前言

柔性制造系统作为一类复杂的人造系统，具有复杂性、递阶结构、不确定性、多目标、多约束、多资源相互协调等特点。鉴于其重要的应用价值和理论意义，相关的分析与控制的研究方法已受到工业界和控制界的广泛关注。制造系统的分析，是基于系统的已知配置和决策，在一定的假设条件下，对系统的各种性能进行评估；而制造系统的控制，则是根据一定的目标和约束来确定系统的配置和决策。研究控制策略的目的是提高系统的性能，而系统分析的目的则是更好地优化和控制系统。制造系统的分析和控制方法，从理论体系上可分为运筹学方法、马尔可夫链、排队论、极大极小代数、摄动分析法、Petri 网、自动机/形式语言和仿真方法等。它们涉及系统的逻辑层次、时间层次和随机层次，进而形成离散事件动态系统的理论框架。

生产调度是制造系统的一个研究热点，也是理论研究中最为困难的问题之一。调度的任务是根据生产目标和约束，为每个加工对象确定具体的加工路径、时间、机器和操作等。优良的调度策略对于提高生产系统的最优性、提高经济效益，有着极大的作用。由于系统建模方法的多样性，以及问题的侧重点不同，调度方法和研究对象也有明显的不同。就对象而言，有确定性和随机性调度、离散事件和连续事件调度、静态和动态调度等。就调度方法而论，有 Gantt 图法、构造型方法、动态规划、分支定界法、排队论方法、规则调度方法和仿真方法等。调度的优化指标包括正规性能指标和非正规性能指标，常用的指标有最大完成时间、平均加工时间、平均延迟时间、生产成本和 E/T 指标等。

目前，对调度理论的研究已受到广泛的关注，并取得了较大的进展，但还很不成熟。其中，对调度问题的复杂性研究已成为工程背景很强的一个应用数学分支。在算法研究方面，基于知识的方法和算法技术相结合的趋势正变得日趋显著，概率分析方法在算法效率和性能方面的研究日益增多。对于难以求得最优解的问题，给出多项式时间的搜索方法具有很大的现实意义，同样，算法的随机性能分析也是比较有效的分析手段。算法研究中，最优化性能的渐近性分析具有理论指导性，而基于启发式算法的误差估计来确定次优度则无疑同样具有很大的意义。由于约束条件的存在导致难以建立数学模型，纯数学的方法往往不易奏效，也不易处理并发现象，因此，从工程中“满意”即可的实际需求出发，寻求满足约束条件的快速有效的优化算法正变得更有现实意义。迄今，计算复杂性理论表明，多数调度问题都属于 NP 难题，目标解的搜索涉及解空间的组合爆炸。线性规划、动态规划、分支定界和梯度下降等传统方法，或是需要目标函数的特殊信息，或是复杂度大，或是优化性能差，因而一般只能处

理小规模问题，难以高效高质量地求解复杂问题。

人们正是由于意识到基于计算和数值式的优化技术的弱点，以及调度问题的约束性、非线性、多极小性、不确定性、大规模性、多目标性等复杂性，才努力研究和发展了统计式全局搜索技术和人工智能的方法，例如模拟退火、遗传算法、禁忌搜索、进化规划、进化策略、神经网络方法、Lagrangian 松弛方法和混沌优化等。这些优化算法通过模拟或揭示某些自然现象、过程和规律而得到发展，以人类认识和理解客观事物的知识、经验等作为解决问题的方法，其思想和内容涉及数学、物理学、生物进化、人工智能、神经科学和统计力学等，为解决复杂问题提供了新的思路和手段。迄今，这些算法独特的优点、机制及其非凡的优化能力，引起了国内外学者的广泛重视，并掀起了该领域的研究热潮，而且在诸多领域得到了成功应用，较满意地解决了一大批传统优化方法难以解决的复杂问题。为此，国际上已设立了相应的学术协会和诸多相关的学术期刊与会议，而遗传算法则是其中研究与应用最广的一类优化算法。

鉴于国际上生产调度和智能优化方法的研究热潮，本书主要介绍各种典型静态调度问题及其遗传算法的设计与实现。全书主要由 5 章组成，内容自成体系。第 1 章介绍调度问题与计算复杂性，包括对工件加工数据和特征、机器加工环境、加工性能指标的描述，进而给出调度问题的表示和分类，并着重介绍 Job Shop 和 Flow Shop 两类典型调度问题；其次，对现有的调度算法进行分类，并介绍多种常用的邻域搜索算法；最后，对计算复杂性和 NP 等基本概念作简单介绍。第 2 章介绍遗传算法的理论与实现技术，首先阐述遗传算法的基本优化流程、模式定理和隐含并行性；其次介绍遗传算法的收敛性理论，包括算法的马尔可夫链描述、标准算法的收敛性和收敛速度估计，进而介绍一般可测状态空间上遗传算法的收敛性；然后，对遗传算法的编码、适配值函数、算法参数、算法操作、终止条件的设计进行介绍，并阐述遗传算法的改进研究和一种免疫遗传算法；最后介绍并行遗传算法。第 3 章介绍 Job Shop 调度问题及其遗传算法设计，首先对 Job Shop 调度进行描述，并介绍若干典型的调度问题；其次，介绍 Job Shop 调度的多种遗传算法编码设计、算法操作和框架设计；进而，介绍 Job Shop 调度的一种有效混合遗传算法和一类模糊 Job Shop 调度的遗传算法设计；最后，对 Job Shop 调度的遗传算法进行了简要综述。第 4 章介绍 Flow Shop 调度及其遗传算法设计，首先对 Flow Shop 调度及其启发式算法进行介绍，进而介绍了若干典型 Flow Shop 调度问题，然后分别介绍置换 Flow Shop 调度、多目标 Flow Shop 调度、一类批量可变 Flow Shop 调度、模糊 Flow Shop 调度和混合 Flow Shop 调度及其遗传算法设计。第 5 章介绍并行机调度及其遗传算法设计，具体包括最小化最大完成时间的遗传算法、最小化最大加权推迟时间的遗传算法、最小化公共交货期下 E/T 指标的遗传算法和一类带工艺约束并行机调度的遗传算法设计。为读者研究方便，本书的附录给出了 Job Shop 和 Flow Shop 的若干 Benchmark 问题。

本书可作为与生产调度及优化相关专业的师生、研究人员以及工程技术人员的参考书。由于作者水平有限，本书难免有不足之处，诚望读者批评指正。

在本书编写过程中，清华大学自动化系郑大钟教授、王雄教授、赵千川副教授等给予了热心指导和建议，清华大学出版社王一玲老师等给予了大力支持，参与研究的有关学生做了大量工作，在此一并表示衷心的感谢。此外，本书的完成得到了国家自然科学基金（“复杂系统基于计算智能的混合优化理论与方法”(60204008)）、国家攀登计划(970211017)、973 国家基础研究项目(G1998020305，2002CB312200)、清华大学基础研究基金、清华大学骨干人才计划等项目的资助，这里谨致谢忱。

王　凌
清华大学自动化系
Email：wangling@mail.tsinghua.edu.cn
2002 年 9 月于清华园

目　录

第1章 调度问题与计算复杂性

生产调度，即对生产过程进行作业计划，作为一个关键模块，是整个先进生产制造系统实现管理技术、运筹技术、优化技术、自动化与计算机技术发展的核心。有效的调度方法和优化技术的研究与应用，是实现先进制造和提高生产效益的基础和关键。改善生产调度方案，可大大提高生产效益和资源利用率，进而增强企业的竞争能力。生产调度的研究主要可分为建模和调度算法设计两方面，它是一个交叉性研究领域，涉及运筹学、数学、计算机工程、控制工程、工业工程等多个学科。其中，建模主要研究调度模型、调度规则、目标函数等内容；算法主要研究算法设计、算法复杂性、算法收敛性和优化质量等内容。本章主要介绍调度问题的描述、调度指标、调度表示、若干邻域搜索算法和计算复杂性的基本概念，为后续章节介绍针对不同类型调度问题的遗传算法设计做基础。

1.1 调度问题及其描述

1.1.1 调度问题

调度问题通常指对生产过程的作业计划，譬如工件在机器上的加工顺序、生产批量的划分等。就生产方式而言，调度问题可分为开环车间(open shop)型和闭环车间(closed shop)型。开环调度问题，也称加工排序问题，它本质上只研究工件的加工顺序，即订单所要求的产品在所有机器上的加工排序，其中订单均来源于顾客，不考虑库存的设立。闭环调度问题除研究工件的加工顺序外，还涉及各产品批量大小的设置，即在满足生产工艺约束条件下寻找一个调度策略，使得所确定的生产批量和相应的加工顺序下的生产性能指标最优，其中顾客需求的产品均由库存提供，生产任务一般只由产品存储策略来决定。

显然，闭环调度问题较开环调度问题要复杂。鉴于批量大小与排序间的耦合性，寻求批量大小和排序的有效同时处理方案很困难，目前处理闭环问题的常用近似方法是，首先确定批量大小，然后确定加工顺序。

对于 m 台机器(machine) $\{M_1,\cdots,M_m\}$ 对 n 个工件(job) $\{J_1,\cdots,J_n\}$ 的加工过程，所谓调度就是分配各工件在各机器上的加工时间。调度通常用甘特图(Gantt chart)表示，图 1.1.1 为 4 个工件、3 台机器的面向机器的甘特图，而图 1.1.2 则是相应的面向工件的甘特图。

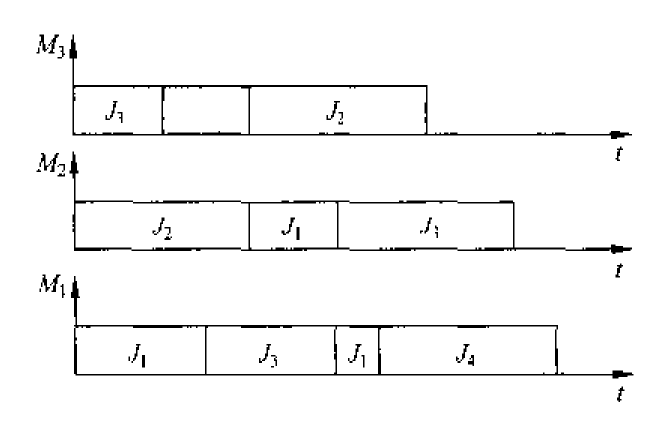


图 1.1.1　4 个工件、3 台机器的面向机器的甘特图

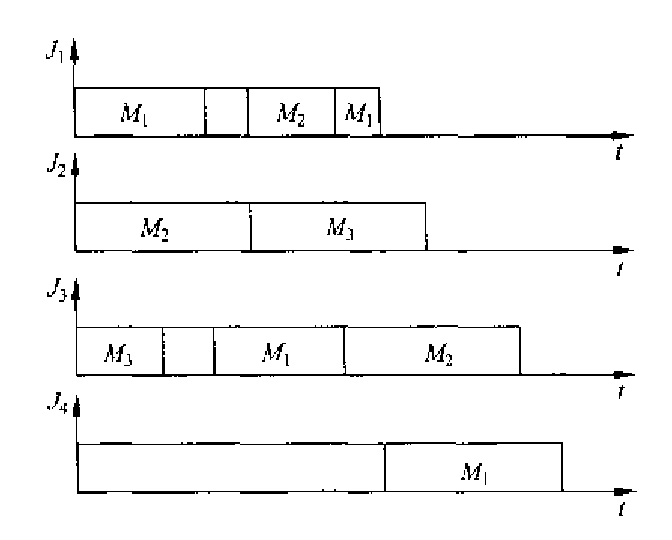


图 1.1.2　4 个工件、3 台机器的面向工件的甘特图

1.1.2　工件加工数据和特性的描述

调度问题中，通常一个工件 J_i 包含 n_i 个操作(operation)$\{O_{i,1},\cdots,O_{i,n_i}\}$，每个操作 O_{ij} 的加工时间或需求为 p_{ij}。若 $n_i=1$，则工件 J_i 仅包含一个操作 O_{i1}，简记其加工时间为 p_i。称工件 J_i 的第 1 个操作可执行的时刻为释放时间或准备时间(release date)，记为 r_i。记加工操作 O_{ij} 的机器集为 $\mu_{ij}\subseteq\{M_1,\cdots,M_m\}$，$O_{ij}$ 可以在 μ_{ij} 中任何一台机器上加工。通常，μ_{ij} 仅对应一台机器或者对应所有机器。前者称为专用机器(dedicated machine)，后者称为并行机(parallel machine)。许多实际生产系统中，各机器可装备相同或不同的工具，操作可以在任何一台装备合适工具的机器上加工，这就是生产系统所谓的柔性，该调度通常称为多目的机器(multi-purpose machines, MPM)调度。若 O_{ij} 的加工过程同时占有 μ_{ij} 中所有机器，则该调度问题称为多处理机任务调度(multi-processor task scheduling)。对于每一工件 J_i，记 t 时刻完成 J_i 的加工费用函数为 $f_i(t)$，记计划完成时间或交货期(due date)为 d_i，与之相关的权

重(weight)为 w_i。

若同一机器上既没有任意两个时间区间重叠，也没有分配给同一个工件的任意两个时间区间重叠，并且满足调度问题的一些特殊工艺约束，则称一个调度为可行(feasible)调度。进而，称使得调度准则或指标最优的可行调度为最优调度(optimal)。

通常，工件加工特性可用六元组 $\beta=\{\beta_1,\beta_2,\beta_3,\beta_4,\beta_5,\beta_6\}$ 来表示，其中

- β_1 用来表示加工方式，包括抢占式(preemption)或非抢占式(non-preemption)。其中，抢占式加工中操作在加工过程中可以被打断，再在原机器或别的机器上重新开始；非抢占式加工则一旦操作开始，直到加工完毕，不能被其他加工打断。通常，记抢占式加工为 $\beta_1=pmtn$，而非抢占式加工则不在 β 中出现 β_1。
- β_2 用来表示工件间的加工优先关系(precedence relation)。该优先关系可以用非循环有向图 $G=(V,A)$ 来表示，其中 $V=\{1,\cdots,n\}$ 表示工件，$(i,k)\in A$，当且仅当 J_i 必须在 J_k 开始加工之前完成。若 G 为任意非循环有向图，则记 $\beta_2=prec$。若 G 对应树(tree)，则记 $\beta_2=tree$。若 G 对应链(chain)，则记 $\beta_2=chain$。若 G 对应的图具有系列-并行(series-parallel)特性，则记 $\beta_2=sp\text{-}graph$。若调度问题不考虑工件加工的优先权，则 β 中不出现 β_2。
- β_3 用来表示工件的准备加工时间。若 $r_i\neq 0$，则记 $\beta_3=r_i$；若所有 $r_i=0$，则 β 中不出现 β_3。
- β_4 表示加工时间或操作数量的限制。若 β_4 记作 $p_i=1$(或 $p_{ij}=1$)，则每个工件(或操作)有 1 次操作需求(如加工时间为 1)。有时，β_4 也可为一些具有明显意义的别的值，如 $p_i\in\{1,2\}$，$d_i=d$。
- β_5 用来表示交货期信息。$\beta_5=d_i$ 表示工件有交货期要求，否则 β 中不出现 β_5。
- β_6 用来表示批量信息，即加工件是否成批联合进行加工。批量调度中，同批的各工件的完成时间等于该批量的完成时间，假定各批量的加工设置时间相同且与加工顺序无关。$\beta_6=p\text{-}batch$ 或 $\beta_6=s\text{-}batch$ 分别表示批量长度等于该批量中所有工件的加工时间的最大值或加工时间之和。若不考虑批量调度，则 β 中不出现 β_6。

1.1.3 机器加工环境的描述

本小节对调度问题的机器加工环境作说明。机器加工环境通常可用一个二参数串 $\alpha=\alpha_1\alpha_2$ 来表示，其中 $\alpha_1\in\{o,P,Q,R,PMPM,QMPM,G,X,O,J,F\}$，符号 o 表示空。若 $\alpha_1=o$，则 $\alpha=\alpha_2$。若 $\alpha_1\in\{o,P,Q,R,PMPM,QMPM\}$，则每个工件仅包含一个操作。

若 $\alpha_1=o$，则每个工件必须在一个规定机器上加工。

若 $\alpha_1 \in \{P,Q,R\}$，则表示并行机加工环境，每个工件可以在 $\{M_1,\cdots,M_m\}$ 中任一台机器上加工。其中，$\alpha_1=P$ 表示相同并行机(identical parallel machines)，即对任意机器 M_j 满足 $p_{ij}=p_i$；$\alpha_1=Q$ 表示均匀并行机(uniform parallel machines)，即 $p_{ij}=p_i/s_j$，s_j 为机器 M_j 的加工速度；$\alpha_1=R$ 表示不相干的并行机(unrelated parallel machines)，即 $p_{ij}=p_i/s_{ij}$，s_{ij} 为与工件相关的机器 M_j 的加工速度。

若 $\alpha_1=PMPM$ 或 $QMPM$，则表示加工环境对应具有相同或均匀速度的多目的机器。

若 $\alpha_1=\{G,X,O,J,F\}$，则表示多操作模型，即每个工件包含多个操作。同时，所有机器是专用性的，即 μ_{ij} 只有一个元素，且各操作间存在优先顺序。该调度称为一般车间(General Shop)调度，记 $\alpha_1=G$。

对于 Job Shop 调度问题，记 $\alpha_1=J$，假定各操作的优先顺序为 $O_{i1}\to O_{i2}\to\cdots\to O_{in_i}$，$i=1,\cdots,n$，且一般假定 $\mu_{ij}\neq\mu_{i,j-1}$，$j=1,\cdots,n_i-1$。若允许 $\mu_{ij}\neq\mu_{i,j-1}$ 不成立，则称之为可重机器 Job Shop(Job Shop with machine repetition)。对于 Job Shop 问题，结合考虑工件特性，若设置 β_4 为 $n_i\leqslant 2$，则所有工件最多包含 2 个操作。

对于 Flow Shop，记 $\alpha_1=F$，它是 Job Shop 的一个特例，即对 $i=1,\cdots,n$ 和 $j=1,\cdots,m$ 有 $n_i=m$，$\mu_{ij}=\{M_j\}$。若 Flow Shop 中各机器上工件的加工顺序相同，则称之为置换(permutation)Flow Shop。若 Flow Shop 不考虑操作的优先顺序，则称之为 Open Shop，记 $\alpha_1=O$。若问题为 Flow Shop 和 Job Shop 的混合问题，则称之为 Mixed Shop，记 $\alpha_1=X$。

另外，α_2 表示机器数。若机器数给定且已知，则记 α_2 为相应的数值；若机器数给定但任意，则记 $\alpha_2=k$；若机器数任意，则记 $\alpha_2=o$。

1.1.4 加工性能指标的描述

鉴于 Job Shop 的代表性，本节以 n 个工件、m 台机器的 Job Shop 为例(约定调度问题为非抢占式、每个工件或操作不能在同一台机器上多次加工，不考虑工件加工优先权和批量)给出调度性能指标。首先，我们先引入一些相关变量和符号。

(1) w_{ij}：工件 J_i 进行第 j 道操作的等待时间。

(2) w_i：工件 J_i 的总加工等待时间，即 $w_i=\sum_{j=1}^{m}w_{ij}$。

(3) C_i：工件 J_i 的加工完毕时间，则 $C_i=r_i+\sum_{j=1}^{m}(w_{ij}+p_{ij})$。

(4) F_i：工件 J_i 的流经时间(flow-time)，即 $F_i=C_i-r_i$。

(5) L_i：工件 J_i 的推迟完成时间(lateness)，即 $L_i=C_i-d_i$。

(6) T_i：工件 J_i 完成的拖后时间(tardiness)，即 $T_i=\max\{L_i,0\}$。

(7) E_i：工件 J_i 完成的提前时间(earliness)，即 $E_i = \max\{-L_i, 0\}$。

(8) I_i：机器 M_i 的空闲时间，即 $I_i = C_{\max} - \sum_{j=1}^{m_i} p_{ij}$，其中 $C_{\max} = \max\{C_1, \cdots, C_n\}$，$m_i$ 为机器 M_i 上加工的工件总数。

(9) $N_w(t)$：t 时刻处于等待状态的待加工工件数。

(10) $N_p(t)$：t 时刻正在加工的工件数。

(11) $N_c(t)$：t 时刻已加工完毕的工件数。

(12) $N_u(t)$：t 时刻未加工完毕的工件数。

显然，$N_c(t) + N_u(t) = n$，$N_w(t) + N_p(t) = N_u(t)$，$N_u(0) = n$，$N_u(C_{\max}) = 0$。为方便起见，表 $X_{\max} = \max\{X_1, \cdots, X_n\}$，表以工件为下标的变量的均值为 $\bar{X} = \sum_{i=1}^{n} X_i / n$，表以时间为参数的变量的均值为 $\bar{Y} = \int_0^{C_{\max}} Y(t) \mathrm{d}t / C_{\max}$。

基于此，我们给出调度问题的一些典型性能指标。

(1) 基于加工完成时间的性能指标：

- 最大流经时间 $F_{\max}$，总流经时间 $\sum_{i=1}^{n} F_i$，加权流经时间 $\sum_{i=1}^{n} w_i F_i$，平均流经时间 $\bar{F}$。
- 最大完成时间 $C_{\max}$(makespan)，平均完成时间 $\bar{C}$。

(2) 基于交货期的性能指标：

- 平均推迟完成时间 $\bar{L}$，最大推迟完成时间 $L_{\max}$。
- 平均拖后时间 $\bar{T}$，最大拖后时间 $T_{\max}$，总拖后完成时间 $\sum_{i=1}^{n} T_i$。
- 拖后工件个数 n_T(完成时间大于交货期的工件个数)或拖后工件比例 n_T/n。

(3) 基于库存的性能指标：

- 平均待加工工件数 $\bar{N}_w$。
- 平均未完成工件数 $\bar{N}_u$。
- 平均已完成工件数 $\bar{N}_c$。
- 平均正在加工工件数 $\bar{N}_p$。
- 平均机器空闲时间 $\bar{I}$。
- 最大机器空闲时间 $I_{\max}$。

(4) 多目标综合性能指标：

- 流经时间与总拖后时间的综合，如 $\bar{F} + \lambda \sum_{i=1}^{n} T_i$，其中 λ 为权重。
- makespan 与总拖后时间的综合，即 $C_{\max} + \lambda \sum_{i=1}^{n} T_i$。

• E/T 指标，即 $\sum(\alpha_i E_i + \beta_i T_i)$，其中 α_i 和 β_i 为权重。

1.1.5 性能指标的正规性、等价性和活动调度

本节介绍性能指标的正规性、等价性，并引出活动调度等概念。

定义 1.1.1 对于一个调度问题，称性能指标函数 R 是正规的，若对于满足不等式关系 $C_1 \leqslant C'_1, \cdots, C_n \leqslant C'_n$ 的任意两组加工完成时间 $\{C_i\}$ 和 $\{C'_i\}$，对 R 必满足 $R(C_1, \cdots, C_n) \leqslant R(C'_1, \cdots, C'_n)$。

容易验证，前节给出的性能指标中，$\bar{C}, C_{\max}, \bar{F}, F_{\max}, \bar{L}, L_{\max}, \bar{T}, T_{\max}$ 和 n_T 属于正规性能指标。研究表明，对于正规性能指标下的调度问题，其求解过程可以简化。

定义 1.1.2 对于 Job Shop 调度问题，称两个性能指标 A 和 B 是等价的，若性能指标 A 下的最优调度也对应性能指标 B 下的最优调度（称 A 的最优性蕴含了 B 的最优性），且反之亦然（即 B 的最优性蕴含了 A 的最优性）。

容易验证，对于 Job Shop 调度，平均完成时间 $\bar{C}$、平均流经时间 $\bar{F}$、平均推迟完成时间 $\bar{L}$ 和平均等待时间 $\bar{W}$ 是等价的，最大完成时间 $C_{\max}$、平均正在加工工件数 $\bar{N}_p$ 和平均机器空闲时间 I 等价，而最大推迟完成时间 $L_{\max}$ 的最优性蕴含了最大拖后完成时间 $T_{\max}$ 的最优性。

此外，Job Shop 调度中在正规性能指标下还经常涉及三种调度，即活动调度、半活动调度和非延迟调度，其定义如下。

定义 1.1.3 称一个调度为活动调度，如果在不推迟其他操作或破坏优先顺序的条件下，其中没有一个操作可以提前加工。

定义 1.1.4 称一个调度为半活动调度，如果在不改变机器上加工顺序的条件下，其中没有操作可以提前。

定义 1.1.5 称一个调度为非延迟调度，如果至少存在一个工件等待加工时，对应地不存在相应的处于空闲的机器。

上述定义表明，若 Job Shop 处于非活动调度下，则一定可以找到某些操作，使其可以更早加工。当然，有时只能通过改变机器上工件的加工次序来做到。譬如当一个操作在前道工序完成后，可将其插入到同一机器中操作时间比它长而出现时刻比它早的另一个操作之前，也即在那个操作还未开始加工前插入到机器的空闲时间内。显然，通过将非活动调度转化为活动调度，正规性能指标必然有所改善。

三者的包含关系如图 1.1.3 所示。

对于正规调度指标，如最大完成时间，业已证明最优调度必为活动调度。因此，在设计优化算法时，如果将搜索空间限于活动调度集，不仅能保证最优调度的存在，而且能够提高优化效率。但对于 JIT(just in time) 这类非正规性能指标的调度问题，完全有可能其最优调度对应非活动调度。

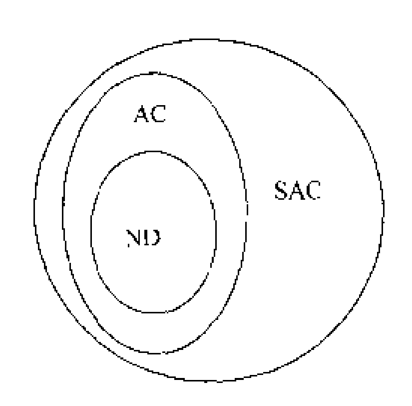


图 1.1.3　三种类型调度的关系

ND 表示非延迟调度集；AC 表示活动调度集；SAC 表示半活动调度集

1.1.6　调度问题的表示

基于 1.1.2 节、1.1.3 节和 1.1.4 节的对工件加工特性、机器加工环境和加工性能指标的描述，可以将任意一个调度问题用 $\alpha/\beta/\gamma$ 来表示，其中 α 表示机器的加工环境，β 表示工件的加工特性，γ 表示加工性能指标。下面举例说明。

例 1.1.1　$P/prec;p_j=1/C_{max}$

该调度问题的属性为：相同并行机加工环境，非抢占式，任意工件加工优先权，准备时间为零，单位加工时间，不考虑交货期和批量，性能指标为最大加工完成时间。图 1.1.4 用有向图给出 2 台机器和 7 个工件的一个例子，图 1.1.5 为该问题的一个可行调度。

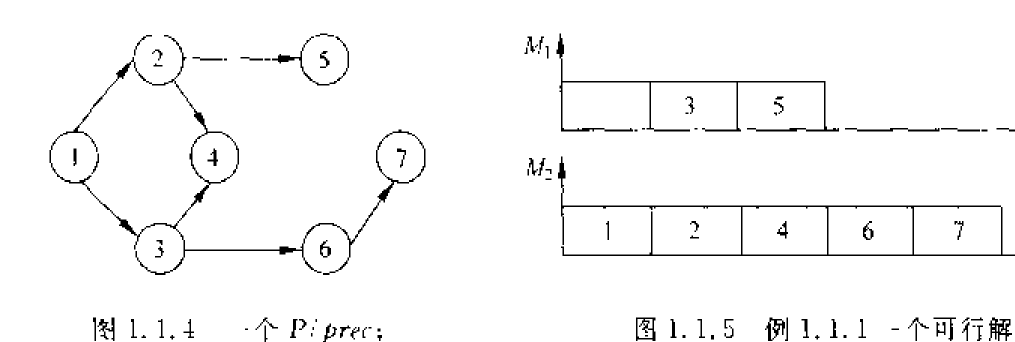


图 1.1.4　一个 $P/prec;p_j=1/C_{max}$ 调度问题示例

图 1.1.5　例 1.1.1 一个可行解

例 1.1.2　$1/r_j;pmpt/L_{max}$

该调度问题的属性为：1 台机器上的抢占式加工，不考虑工件加工优先权，准备时间、操作时间和交货期见表 1.1.1，不考虑批量，性能指标为最大推迟完成时间。图 1.1.6 为该问题的一个调度。

表 1.1.1　例 1.1.2 的加工数据

i	1	2	3	4
p_i	2	1	2	2
r_i	1	2	2	7
d_i	2	3	4	8

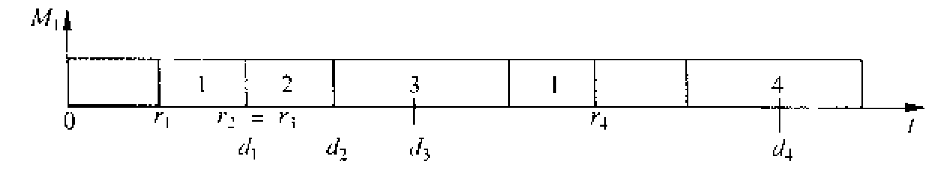


图 1.1.6　例 1.1.2 的一个可行解

例 1.1.3　$J_3/p_{ij}=1/C_{\max}$

该调度问题的属性为:3 台机器的 Job Shop 调度(允许一个工件在同一台机器上多次操作),非抢占式加工,不考虑加工优先权,准备时间为零,各操作需要单位操作时间,不考虑交货期,不考虑批量,性能指标为最大完成时间。表 1.1.2 给出了各操作对应的机器;图 1.1.7 为该问题的一个调度。

表 1.1.2　操作 O_{ij} 对应的机器

i	j			
	1	2	3	4
1	M_1	M_2	M_3	M_1
2	M_2	M_3		
3	M_3	M_1		
4	M_1	M_3	M_1	
5	M_1	M_1	M_2	M_3

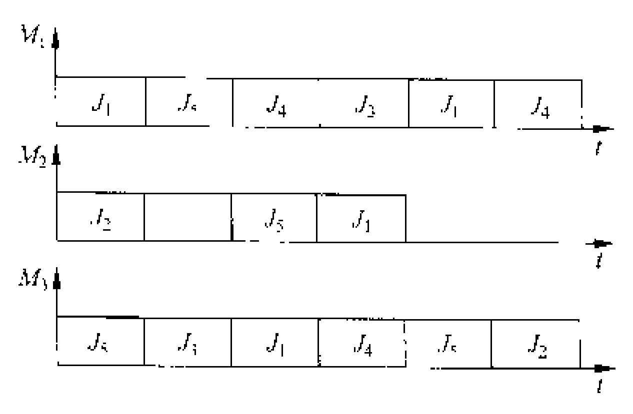


图 1.1.7　例 1.1.3 的一个可行解

例 1.1.4　$1/s\text{-}batch/\sum w_iC_i$

该调度问题的属性为：1 台机器上的非抢占式 *s-batching* 加工问题，操作时间和权值见表 1.1.3，不考虑交货期，考虑批量，性能指标为加权完成时间和。图 1.1.8 为该问题的一个 3 批量调度，其性能指标值等于 $2\times3+(1+1+4)\times10+(1+4)\times15=141$。

表 1.1.3　例 1.1.4 批量长度为 1 时的加工数据

i	1	2	3	4	5	6
p_i	3	2	2	3	1	1
w_i	1	2	1	1	4	4

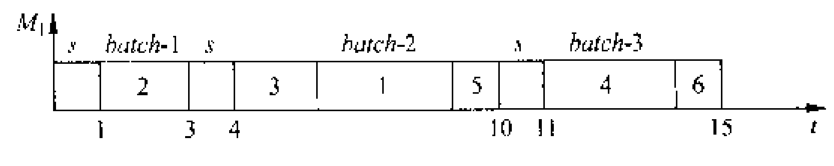


图 1.1.8　例 1.1.4 的一个可行解

1.1.7　Job Shop 和 Flow Shop 调度问题

Job Shop 和 Flow Shop 调度问题是具有特殊工件特性和加工环境的最典型和最重要的调度问题，通常是特殊的开环调度问题。鉴于目前大量文献所讨论的调度问题大都限于这两种调度问题，本节对其单独做介绍。

Job Shop 和 Flow Shop 调度问题研究 n 个工件在 m 台机器上的加工过程，O_{ij} 表示第 i 个工件在第 j 台机器上的操作，相应的操作时间 p_{ij} 为已知，事先给定各工件在各机器上的加工次序（称为技术约束条件），要求确定与技术约束条件相容的各机器上所有工件的加工次序，使加工性能指标达到最优。若各工件的技术约束条件相同，一个 Job Shop 调度问题就转化为较为简单的 Flow Shop 调度问题。进而，若各机器上各工件的加工次序也相同，则问题可进一步转化为置换 Flow Shop 调度问题。

在典型的 Job Shop 调度问题中，除技术约束外，通常还假定以下条件：

- 各工件经过其准备时间后即可开始加工；
- 每一时刻每台机器只能加工一个工件，且每个工件只能被一台机器所加工，同时加工过程为不间断，整个加工过程中机器均有效；
- 整个加工过程中，每个工件不能在同一台机器上加工多次；
- 各工件必须按工艺路线以指定的次序在机器上加工；
- 不考虑工件加工的优先权；
- 操作允许等待，即前一个操作未完成，则后面的操作需要等待；

- 所有机器处理的加工类型均不同；
- 除非特殊说明，工件的加工时间事先给定，且在整个加工过程中保持不变；
- 除非特殊说明，工件加工时间内包含加工设置时间；
- 除非特殊说明，缓冲区容量为无穷大。

显然，这两种调度问题对应特殊的 α 和 β。具体而言，$\alpha_1=J$ 或 F（每个工件只能在同一台机器上最多加工一次，即 μ_{ij} 最多只对应一台机器且 $\mu_{ij}\neq\mu_{i,j+1}$），$\alpha_2=m$，β_1 和 β_2 不出现（即非抢占式且不存在工件加工优先权），$\beta_3=r_i$ 或不出现，β_4 对应的 p_{ij} 由问题给出，$\beta_5=d_i$ 或不出现，β_6 不出现（不考虑批量大小）。不加特殊说明，本书讨论的车间调度问题均满足上述假设。

鉴于 Job Shop 和 Flow Shop 的特殊性、典型性、重要性，通常将其称为基本调度问题，一般用 $n/m/A/B$ 将其简明表示，其中 n 表示工件数，m 表示机器数，A 表示工件流经机器的类型（Job Shop 用 G 表示，Flow Shop 用 F 表示，置换 Flow Shop 用 P 表示），B 表示性能指标（如 $C_{\max}$，$L_{\max}$，F 等）。为简单起见，对于满足上述假设的 Job Shop 或 Flow Shop 调度问题，本书后续章节用 $n/m/A/B$ 替代 $\alpha/\beta/\gamma$ 加以表示。

1.2 调度算法分类与邻域搜索算法

求解调度问题的方法统称为调度优化算法，它可区分为精确求解方法和近似求解方法。其中精确求解方法包括解析方法、穷举方法、分支定界等；近似求解方法包括基于规则的构造方法、邻域搜索方法等。下面，首先对调度算法进行简单介绍和分类，进而介绍邻域搜索方法流程和若干改进的邻域搜索方法。

1.2.1 调度算法分类

所谓优化算法，其实就是一种搜索过程或规则，它是基于某种思想和机制，通过一定的途径或规则来得到满足用户要求的问题的解。

调度问题的求解方法通常可分为解析方法、枚举方法、构造性方法、邻域搜索方法、人工智能方法等。

(1) 解析方法。指针对简单小规模调度问题的解析求解方法。

(2) 枚举方法（enumerative methods）。该类方法对小规模问题比较有效，但对大规模问题计算量和存储量难以承受。

- 分支定界（bound and branch）。
- 数学方法（mathematical method）。譬如整数规划、混合整数规划、分解方法、代理对偶方法、Lagrangian 松弛方法等。

(3) 构造性方法（constructive method）。该类方法能够快速建立问题的解，但通常解质量较差，否则需要建立复杂的启发式规则。

- 优先分配规则(priority dispatch rule)。譬如 SPT,LRM,MWR 等。
- 基于瓶颈的启发式方法(bottleneck based heuristics)。如移动瓶颈过程(shifting bottleneck procedure),beam 搜索等。
- 插入方法(insertion algorithm)。如 CDS 法、NEH 法等。

(4) 邻域搜索算法(local search method)。该类方法从若干解出发,对其邻域的不断搜索和当前解的替换来实现优化。

- 迭代改进法(iterative improvement)。
- 模拟退火算法(simulated annealing, SA)。
- 进化计算(evolutionary computation, EC)。它包括遗传算法(genetic algorithm, GA)、进化规划(evolutionary programming, EP)、进化策略(evolutionary strategy, ES)和遗传编程(genetic programming, GP)。
- 禁忌搜索(tabu search, TS)。
- 巢分区(nested partition, NP)。
- 变邻域搜索(variable neighborhood search, VNS)。
- 噪声方法(noising method, NM)。
- 阈值接受法(threshold accepting, TA)。
- 可变深度搜索(variable deep search, VDS)。

(5) 人工智能方法(artificial intelligence)。该类方法利用人工智能的原理和技术进行搜索,譬如将优化过程转化为智能系统动态的演化过程,基于系统动态的演化来实现优化。

- 神经网络(neural network, NN)。
- 专家系统(expert system, ES)。
- 蚁群系统(ant system, AS)。
- 混沌搜索(chaotic search, CS)。
- 免疫算法(immunity algorithm)。

另外,还有上述各算法的混合方法。

当然,我们还可以从别的角度对优化算法进行分类,譬如确定性算法和不确定性算法、局部优化算法和全局优化算法等,在此不再展开。需要指出的是,大量实际调度问题存在不确定性,目标函数的评价对仿真有依赖性,而且搜索空间组合爆炸;对于这类所谓的随机仿真优化问题,研究相应的有效仿真优化算法近年来已成为国际学术界的前沿研究课题(王凌等, 2003)。在此推荐序优化(ordinal optimization, OO)、摄动分析(perturbation analysis, PA)等方法,它们可有效处理调度问题的一些随机因素,但本书不予展开介绍。由于本书主要介绍调度问题的遗传算法设计,而遗传算法是一种现代启发式方法,可归入邻域搜索算法类(王凌, 2001),因此以下着重对邻域搜索算法做介绍。

1.2.2 邻域搜索算法

邻域函数是优化中的一个重要概念，其作用就是指导如何由一个(组)解来产生一个(组)新的解。邻域函数的设计往往依赖于问题的特性和解的表达方式(编码)。由于优化状态表征方式的不同，函数优化与组合优化中的邻域函数的具体方式将明显存在差异。

函数优化中邻域函数的概念比较直观，利用距离的概念通过附加扰动来构造邻域函数是最常用的方式，如 $x'=x+\eta\cdot\xi$，其中 x' 为新解，x 为旧解，η 为尺度参数，ξ 为满足某种概率分布的随机数、白噪声、混沌序列或梯度信息等。显然，采用不同的概率分布(如高斯分布、柯西分布、均匀分布等)或下降策略，将实现不同性质的状态转移。但在组合优化中，传统的距离概念显然不再适用，然而其基本思想仍旧是通过一个解产生另一个解。下面对邻域函数给出一个一般性定义，并以旅行商问题为例进行解释。

定义 1.2.1 令 (S,F,f) 为一个组合优化问题，其中 S 为所有解构成的状态空间，F 为 S 上的可行域，f 为目标函数，则一个邻域函数可定义为一种映射，即 $N: S\to 2^S$。其涵义是，对于每个解 $i\in S$，一些"邻近"i 的解构成 i 的邻域 $S_i\subset S$，而任意 $j\in S_i$ 称为 i 的邻域解或邻居。通常约定，$j\in S_i\Leftrightarrow i\in S_j$。

旅行商问题(traveling salesman problem, TSP)是典型的组合优化问题，该问题可描述为：给定 n 个城市和两两城市间的距离，要求确定一条经过各城市当且仅当一次的最短路线。其图论描述为：给定图 $G=(V,A)$，其中 V 为顶点集，A 为各顶点相互连接组成的边集，已知各顶点间的连接距离，要求确定一条长度最短的 Hamilton 回路，即遍历所有顶点当且仅当一次的最短回路。通常，TSP 问题的解可用置换排列来表示，如排列(1, 2, 3, 4)可表示 4 个城市 TSP 的一个解，即旅行顺序为 1,2,3,4。那么，k 个点的交换就可认为是一种邻域函数。譬如，不考虑由解的方向性和循环性引起的重复性，上述排列的 2 点交换对应的邻域函数将产生新解(2,1,3,4)，(3,2,1,4)，(4,2,3,1)，(1,3,2,4)，(1,4,3,2)，(1,2,4,3)。

基于邻域函数的概念，就可以对局部极小和全局最小进行定义。

定义 1.2.2 若 $\forall j\in S_i\cap F$，满足 $f(j)\geq f(i)$，则称 i 为 f 在 F 上的局部极小解；若 $\forall j\in F$，满足 $f(j)\geq f(i)$，则称 i 为 f 在 F 上的全局最小解。

局部搜索算法是基于贪婪接受的思想利用邻域函数进行搜索的，它通常可描述为：算法从一个初始解出发，利用邻域函数持续地在当前解的邻域中搜索比它好的解，若能够找到如此的解，就以之成为新的当前解，然后重复上述过程，否则结束搜索过程，并以当前解作为最终解。

[**贪婪性局部搜索算法**]：

步骤 1：在解空间 S 中随机选择一个解 s。

步骤 2：在解 s 的邻域 $N(s)$ 中产生一个邻域解 s'。

步骤 3：若 $f(s')<f(s)$ 则令 $s=s'$，否则保持当前解为 s。

步骤 4：若算法终止条件满足则结束搜索并输出解 s，否则转步骤 2。

可见，局部搜索算法尽管具有通用易实现的特点，但搜索性能完全依赖于邻域函数和初始解，邻域函数设计不当或初值选取不合适，则算法最终的性能将会很差。同时，贪婪思想无疑将使算法丧失全局优化的能力，也即算法在搜索过程中无法避免陷入局部极小。因此，若不在搜索策略上进行改进，那么要实现全局优化，局部搜索算法采用的邻域函数必须是“完全的”，即邻域函数将导致解的完全枚举，而这在大多数情况下是无法实现的，而且穷举的方法对于大规模问题在搜索时间上是不允许的。

鉴于局部搜索算法的上述缺点，近年来进化计算、模拟退火算法、禁忌搜索、噪声方法、混沌搜索、变邻域搜索、巢分区、隧道法等改进型邻域搜索算法在组合优化领域，尤其是在生产调度领域得到了广泛的研究与应用。它们从不同的角度利用不同的搜索机制和策略来实现对局部搜索算法的改进，以取得较好的全局优化性能。下面对几种常用的改进型邻域搜索方法的原理和步骤做简单介绍。

1. 进化计算(EC)

EC 是遗传算法(GA)、进化规划(EP)、进化策略(ES)和遗传编程(GP)的统称，是基于“适者生存”的一类高度并行、随机和自适应优化算法，但各算法在遗传操作上有细微差别。以 GA 为例，它将问题的求解表示成“染色体”的适者生存过程，通过“染色体”群的一代代不断进化，包括复制、交叉和变异，最终收敛到“最适应环境”的个体，从而求得问题的最优解。其中，复制通常采用比例复制，即复制概率正比于个体的适配值，即意味着适配值高的个体在下一代中复制自身的概率大，从而提高了种群的平均适配值；交叉通过交换两父代个体的部分信息构成后代个体，使得后代继承父代的有效模式，从而有助于产生优良个体；变异通过随机改变个体中某些基因而产生新个体，有助于增加种群的多样性，避免早熟收敛。

[**基本进化计算**]：

步骤 1：在解空间中随机产生一组解构成初始种群并评价当前种群。

步骤 2：若算法终止条件满足则结束搜索并输出搜索解，否则继续以下步骤。

步骤 3：在当前父代种群中选择个体并通过遗传操作(如交叉、变异)得到一组新个体。

步骤 4：评价新个体。

步骤 5：按某种替换方式得到下一代种群。

区别于传统优化算法，EC 的特点在于：对问题参数编码成“染色体”后进行进化操作，而不针对参数本身，从而不受函数约束条件的限制，如连续性、可导性等；搜索过程从问题解的一个集合开始，而不是单个个体，具有隐含并行搜索特性，可大大减

小陷入局部极小的可能；遗传操作具有随机性，并根据个体的适配值信息进行搜索，而无需其他信息。其优越性则主要表现在：算法进行全空间并行搜索，并将搜索重点集中于性能高的部分，从而能提高效率且不易陷入局部极小；具有固有的并行性，通过对种群的遗传处理可处理大量的模式，且容易并行实现。因此，EC从局部极小到全局最优是通过群体化的概率性遗传操作来实现的。目前，EC是应用最广的智能优化算法，但在避免早熟收敛方面还有待深入研究。

2. 模拟退火(SA)

SA最早由Kirkpatrick等(1983)用于组合优化，它是基于Monte Carlo迭代求解的一种通用随机寻优算法。其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性，通过设定一初温和初态，伴随温度的不断下降，结合概率突跳特性在解空间中通过邻域函数进行随机搜索，最终得到全局最优。

[**基本模拟退火算法**]：

步骤1：在解空间S中随机产生一个解s，并令$i=0$，最优解$s^*=s$，最优值为$f(s^*)$，确定初始温度t_i。

步骤2：若算法终止条件满足则结束搜索并输出最优解s^*，否则继续以下步骤。

步骤3：在当前解s的邻域$N(s)$中产生邻域解s'。

步骤4：若$f(s')<f(s^*)$，则令$s^*=s'$，$f(s^*)=f(s')$。

步骤5：若$\xi\in[0,1]<\min\{1,\exp(-\Delta/t_i)\}$，则令$s=s'$，$f(s)=f(s')$，其中$\Delta=f(s')-f(s)$。

步骤6：若抽样稳定准则满足则转步骤7，否则转步骤3。

步骤7：降温，即令$t_{i+1}=update(t_i)$，并令$i=i+1$，再转步骤2。

区别于传统邻域搜索算法，SA以概率$\min\{1,\exp(-\Delta/t)\}$来接受新状态，其中$\Delta$为新旧状态的目标值差，$t$为温度。显然，算法保留了向优良解概率1的趋化过程，而赋予差的状态一定的可控接受概率，同时这种概率随温度的不断下降而趋于零。正是由于这种概率性的劣向转移，使得算法具有跳出局部极小而实现全局最优的能力。迄今，SA得到了广泛应用，但在提高优化效率方面还有待进一步研究。

3. 禁忌搜索(TS)

TS最早由Glover(1982)提出，它是对局部邻域搜索的一种扩展，是一种全局逐步寻优算法，是对人类智力过程的一种模拟。TS一方面沿用局部邻域搜索的思想，用于实现邻域搜索；另一方面通过引入一个灵活的存储结构和相应的禁忌准则来设置禁忌表和禁忌对象，通过标记对应已搜索到的局部最优解的一些对象，并在进一步的迭代搜索中尽量避开这些对象(而不是绝对禁止循环)，从而体现算法避免迂回搜索的特点，并保证对不同的有效搜索途径的探索；另外，通过设置藐视准则来奖励和赦免一些优良状态(譬如判断状态是否优于“best so far”状态)。

[基本禁忌搜索算法]：

步骤1：给定算法参数，随机产生初始解 x，置禁忌表为空。

步骤2：判断算法终止条件是否满足？若是，则结束算法并输出优化结果；否则，继续以下步骤。

步骤3：利用当前解 x 的邻域函数产生其所有(或若干)邻域解，并从中确定若干候选解。

步骤4：对候选解判断藐视准则是否满足？若成立，则用满足藐视准则的最佳状态 y 替代 x 成为新的当前解，即 $x=y$，并用与 y 对应的禁忌对象替换最早进入禁忌表的禁忌对象，同时用 y 替换"best so far"状态，然后转步骤6；否则，继续以下步骤。

步骤5：判断候选解对应的各对象的禁忌属性，选择候选解集中非禁忌对象对应的最佳状态为新的当前解，同时用与之对应的禁忌对象替换最早进入禁忌表的禁忌对象元素。

步骤6：转步骤2。

区别于传统邻域搜索算法，TS的主要特点是：搜索过程中可以接受劣解，因此具有较强的"爬山"能力；新解不是在当前解的邻域中随机产生，而或是优于"best so far"的解，或是非禁忌的最佳解，因此选取优良解的概率远远大于其他解。尽量避免迂回搜索，允许接受邻域中非禁忌的劣解和满足藐视准则的优良解，这是TS实现高效搜索的关键。由于TS类似SA同为串行搜索，提高优化效率也是TS面临的重要课题。

4. 噪声方法(NM)

NM方法是Charon等(2001)最近提出的一种组合优化算法，它可理解为SA的一种变型或推广。区别于传统邻域搜索算法和SA，NM对新旧状态的目标值差 Δ 附加一定的扰动，即 $\Delta_n=\Delta+\rho$，其中 $\rho\in[-r,r]$ 为扰动，然后判断 $\Delta_n<0$ 是否成立，成立则接受新状态，否则保留旧状态。当上述搜索过程持续一定步数后，算法适当降低 r 值来减小噪声幅度，显然当 $r=0$ 时NM成为传统的趋化性下降搜索，相对于SA的 $t=0$。

对于SA而言，若 $\Delta<0$ 则新状态一定接受，若 $\Delta\geqslant0$ 则以概率 $\exp(-\Delta/t)$ 接受新状态；而对于NM而言，若 $\Delta<-r$ 则一定接受新状态，若 $\Delta>r$ 则保留旧状态，若 $\Delta\in[-r,r]$ 则新状态将以一定概率接收。由此可见，NM允许概率性地抛弃一些性能增长幅度不大的优良解，而完全不接受性能过劣的解，这可认为是对SA突跳能力的扩充。目前，NM方法在诸多组合优化问题上得到成功应用，但其收敛性和收敛速度，尤其是初始扰动区间、扰动下降幅度等方面还缺乏理论成果。

5. 混沌搜索(CS)

混沌是一种普遍的非线性现象，其行为复杂且类似随机，但存在精致的内在规律

性。混沌具有其独特性质：随机性，即具有类似随机变量的杂乱表现；遍历性，即能不重复地历经一定范围内的所有状态；规律性，即由确定性迭代式产生。介于确定性和随机性之间，混沌具有丰富的时空动态，系统动态的演变可导致吸引子的转移。最重要的是，借鉴混沌动态的遍历性，搜索过程不受能量障碍的限制，从而可有效避免优化过程陷入局部极小解，这与SA的概率性劣向转移和TS的禁忌表检验存在明显区别。因此，混沌已成为一种新颖的优化工具，并受到广泛重视和大量研究（王凌等，2001）。

对于组合问题，混沌通常与Hopfield神经网络（HNN）结合使用。一方面，通过在HNN中引入外部机制产生的混沌噪声来构成混沌神经网络，基于混沌时间序列永不可精确重复的规律，当外部引入的混沌的幅度足够小时，能量函数曲面上的寻优过程在具有丰富时空特性的混沌动态的激励下能够产生避免陷入局部极小的能力，并保持对问题的原始描述不变。另一方面，混沌内在嵌入在HNN中，可通过控制参数来实现混沌搜索到确定性搜索的转换，即先利用瞬时混沌动态进行全局遍历性搜索，而后再转入梯度下降搜索。

轨道遍历性，即混沌序列能够不重复地历经一定范围内的所有状态，是混沌用于函数优化的根本出发点。通常，基于混沌动态的搜索过程分为两个阶段：首先，基于确定性迭代式产生的遍历性轨道对整个解空间进行考察，当满足一定终止条件时，认为搜索过程中发现的最佳状态已接近问题的最优解（只要遍历性搜索轨道足够长，这种情况总能实现），并以此作为第二阶段的搜索起始点；其次，以第一阶段得到的结果为中心，通过附加小幅度的扰动进一步进行局部区域内的细搜索，直至算法终止准则满足，而附加的扰动可以是多样化的。归根到底，混沌搜索的本质是借助遍历性来避免陷入局部极小的。因此，如何利用和控制好混沌动态是这类算法产生良好性能的关键。

6. 变邻域搜索（VNS）

VNS方法是Hansen等（2001）提出的又一种meta-heuristic算法，最近又有若干改进方法并成功应用于若干典型NP-hard问题。区别于传统方法的单一邻域结构，VNS首先定义一系列邻域结构，并在搜索过程中系统性地不断改变（扩大）邻域搜索范围，直到有新的更好的解产生。

［**基本变邻域搜索算法**］：

步骤1：定义一系列邻域结构 N_k，$k=1,\cdots,k_{max}$，给定一个初始解。

步骤2：重复下列过程直至算法终止：

(2.1) 令 $k=1$。

(2.2) 重复下列过程直至 $k=k_{max}$：

(2.2.1) 由当前解 x 在其第 k 种邻域 N_k 内随机产生新解 x'；

(2.2.2) 以 x' 为初值应用邻域搜索方法得到局部极小解 x''；

(2.2.3) 若 x'' 优于 x，则 $x=x''$ 并令 $k=1$，否则 $k=k+1$。

可见，确定邻域结构及其数量，确定各邻域结构在搜索中应用的次序及其改变的策略，是 VNS 要解决的基本问题。例如旅行商问题，可以定义邻域搜索结构为 k-OPT 变换，$k=2,3,\cdots,n$，即两条不同回路间仅存在 k 条不同的边。由于 k-OPT 的解必然优于 $(k-1)$-OPT 的解，因此通过增大 k 即可扩大邻域搜索范围，并能取得更好的解。需要指出的是：VNS 除了多邻域的应用，也可采用多种不同的子邻域搜索方法；另外，研究问题目标函数中局部极小和相应波峰波谷的拓扑结构不仅有助于对问题有更清晰的了解，而且有助于提供 VNS 进行搜索的有用信息，但迄今这方面的研究成果相当少，且限于极少数几个问题。

7. 巢分区方法（NP）

NP 方法是由 Shi 等（1999）最近提出的一种全局优化算法，其基本思想是对可行域进行系统性分区，然后集中搜索优良解可能位于的区域。在每一步迭代中，算法跟踪最有希望的区域，并用随机抽样得到的信息来实现有希望区域间的转移。在第 k 步迭代中算法进行如下操作：

[**巢分区方法的第 k 步迭代流程**]：

步骤 1：分区。将最有希望的区域 $\sigma(k)$ 分成 $m_{\sigma(k)}$ 个子区域 $\sigma_1(k),\cdots,\sigma_{m_{\sigma(k)}}(k)$，并将整个解空间中余下的空间 $\Theta\backslash\sigma(k)$ 合并为一个区域 $\sigma_{m_{\sigma(k)}+1}(k)$。

步骤 2：随机抽样。在每个区域 $\sigma_j(k)$，$j=1,\cdots,m_{\sigma(k)}+1$ 中随机选取 n_j 个点 $\theta_1^j,\cdots,\theta_{n_j}^j$，并计算其目标值 $f(\theta_i^j)$，$i=1,\cdots,n_j$。

步骤 3：计算希望指标。给定一个希望指标函数 $I:\sum\rightarrow R$，计算每一区域 $\sigma_j(k)$ 的希望指标。譬如定义区域的最佳性能为 $I(\sigma_j(k))=\min\limits_{\theta\in\sigma_j(k)}f(\theta)$，其估计值为 $\hat{I}(\sigma_j(k))=\min\limits_{i=1,\cdots,n_j}f(\theta_i^j)$。

步骤 4：返回。确定最佳子区域 $j_k=\arg\min\hat{I}(\sigma_j(k))$，若 $j_k\neq m_{\sigma(k)}+1$ 则将该区域作为下一步迭代的最有希望区域，否则用设计好的撤退规则返回到某个区域。

NP 算法保证每一步对所有可行空间均进行抽样，尤其对最有希望区域通过重复分区而进行重点抽样，如此逐步缩小最佳区域。如何分区、如何得到抽样点、如何定义指标函数、如何选择撤退规则、如何选择初始最佳区域，这些都是该方法的基本问题。该方法非常容易实施，在理论上具有概率 1 全局收敛特性，并且具有自然的并行性，因此也可在并行计算机上实施。

8. 隧道方法（TM）

鉴于传统优化算法搜索效率受初始点和优化曲面的影响，隧道方法（Roychowdhury 等，2000）是又一种脱离局部极小或平坦区的有效方法，即"钻地

道”。TM方法通常是和局部搜索方法结合使用的，迄今在组合优化和函数优化方面均有较成功的应用。不失一般性，以一维函数优化问题为例，假设 x_1^* 为梯度下降法达到的解，通过附加在其邻域内产生点 $x_1^*+\epsilon$，则由于 x_1^* 为局部极小解，故其性能必然优于 $x_1^*+\epsilon$，进而造成搜索陷入局部极小。TM的基本思想是排除所有劣于当前状态的局部极小的访问，而转到另一个吸引域的高点，进而再一次由下降法得到新的局部极小，如此重复则可得到全局最优。TM的难点在于如何实现“隧道”的作用，而这通常因问题类型而异且需对问题信息有足够了解，因此其应用也受到一定的局限性。

当然，目前有效的邻域搜索算法远不限于以上介绍的几种，在此不予一一展开，读者可参阅相应文献。

1.3 计算复杂性与NP完全问题

1.3.1 计算复杂性基本概念

对于对称旅行商问题，所有可行路径有 $(n-1)!/2$ 条，若以路径比较为基本操作，则需 $(n-1)!/2-1$ 次基本操作才能够保证得到绝对最优解。对于每秒执行一百万次操作的计算机，当 $n=20$ 时就需1 929年才能找到最优解。因而，显然对较大规模问题穷举比较是不切合实际的。也就是说，有些优化算法所需的计算时间和存储空间是难以承受的，因此算法可解的问题在实践中并不一定可解。鉴于许多调度问题的求解复杂性远远大于旅行商问题，所以我们必须对计算复杂性理论有所了解，这也是最优化的理论基础。只有了解所研究问题的复杂性，才能有针对性地设计算法，进而提高优化效率。

算法的时间和空间复杂性对计算机的求解能力有重大影响。算法对时间和空间的需要量称为算法的时间复杂性和空间复杂性。问题的时间复杂性是指求解该问题的所有算法中时间复杂性最小的算法的时间复杂性，问题的空间复杂性也可类似定义。按照计算复杂性理论研究问题求解的难易程度，可把问题分为P类、NP类和NP完全类。

算法或问题的复杂性一般表示为问题规模 n（如TSP问题中的城市数）的函数，时间复杂性记为 $T(n)$，空间复杂性记为 $S(n)$。在算法分析和设计中，沿用实用性的复杂性概念，即把求解问题的关键操作（如加、减、乘、比较等运算）指定为基本操作，而把算法执行基本操作的次数定义为算法的时间复杂性，算法执行期间占用的存储单元则定义为算法的空间复杂性。在分析复杂性时，可以求出算法的复杂性函数 $p(n)$，也可用复杂性函数主要项的阶 $O(p(n))$ 来表示。若算法 A 的时间复杂性为 $T_A(n)=O(p(n))$，且 $p(n)$ 为 n 的多项式函数，则称算法 A 为多项式算法，而把时间

复杂性大于多项式时间的算法统称为指数时间算法。

1.3.2 P,NP,NP-C,NP-hard

P类问题指具有多项式时间求解算法的问题类。但迄今为止,许多优化问题仍没有找到求得最优解的多项式时间算法,通常称这种比P类问题更广泛的问题为非多项式确定问题,即NP问题。NP的概念通常由判定问题引入,下面介绍相应的若干概念。

定义1.3.1 实例是问题的特殊表现,所谓实例就是确定了描述问题特性的所有参数的问题,其中参数值称为数据,这些数据占用计算机的空间称为实例的输入长度。

定义1.3.2 若一个问题的每个实例只有"是"或"否"两种回答,则称该问题为判定问题,并称肯定回答的实例为"是"实例,否定回答的实例为"否"实例或非"是"实例。

定义1.3.3 若存在一个多项式函数$g(x)$和一个验证算法H,对一类判定问题A的任何一个"是"的判定实例I都存在一个字符串S是I的"是"回答,满足其输入长度$d(S)$不超过$g(d(I))$,其中$d(I)$为I的输入长度,且验证算法验证S为I的"是"回答的计算时间不超过$g(d(I))$,则称判定问题A为非多项式确定问题,简称NP。

由此可见,判定问题是否属于NP的关键是对"是"的判定实例是否存在满足上述条件的一个字符串和算法,其中字符串在此可理解为问题的一个解,而定义中没有强调字符串和算法是如何得到的。可见,$P\subset NP$。然而,是否$P=NP$? 这个问题至今没有答案!

归约和转换是描述问题特性的常用方法。如果能够将几类问题归结为一个问题,则一旦解决了一个归结后的问题,其他几类问题也就解决了。

定义1.3.4 给定问题A_1和A_2,称A_1可多项式转换为A_2,若存在一个多项式函数$g(x)$和一个字符串,满足:

(1) 对A_1的任何一个实例I_1,在其输入长度的多项式时间$g(d(I_1))$内构造A_2的一个实例I_2,使其长度不超过$g(d(I_1))$;

(2) 由此构造使得实例I_1和I_2的解一一对应,且d_1为I_1的"是"回答的充要条件为d_1对应的解是I_2的一个"是"回答。

定义1.3.5 给定判定问题A_1和A_2,称A_1可多项式归约为A_2,若存在多项式函数$g_1(x)$和$g_2(x)$,使得对A_1的任何一个实例I,在多项式时间内构造A_2的一个实例,其输入长度不超过$g_1(d(I))$,并对A_2的任何一个算法H_2,都存在问题A_1的一个算法H_1,使得H_1调用H_2且计算时间$f_{H_1}(d(I))\leqslant g_2(f_{H_2}(g_1(d(I))))$。

由此可见,若问题A_2存在多项式时间算法,则问题A_1一定存在多项式时间算

法。若问题 A_1 可多项式转换为问题 A_2,对 A_2 的一个算法 H_2,可按如下步骤构造 A_1 的算法。

[**问题 A_1 的算法构造**]:

步骤1:对问题 A_1 的任何一个实例 I_1,先用多项式时间 $g_1(d(I_1))$ 构造问题 A_2 的一个实例 I_2。

步骤2:调用算法 H_2 求解 I_2,此步计算时间为 $f_{H_2}(g_1(d(I_1)))$。

如此构造的算法对问题 A_1 的任何一个实例 I_1 的计算时间不超过 $g_1(d(I_1))+f_{H_2}(g_1(d(I_1)))$。

进一步,我们给出 NP-C 和 NP-hard 的概念。

定义 1.3.6 称判定问题 $A\in$ NP-C(NP-Complete),若 $A\in$ NP 且 NP 中的任何一个问题可多项式归约为问题 A。称问题 A 为 NP-hard,只要上述第二个条件成立。

由此可见,NP-C$\subset$NP-hard,而两者的区别仅在于 NP-C 必须判断问题属于 NP 类。同时,若我们已知一个问题为 NP-C 或 NP-hard,当遇到一个新问题时,若已知问题可多项式归约为新问题,则新问题为 NP-hard,进而若可验证新问题属于 NP 类,则新问题为 NP-C。

为直观起见,上述四类问题的关系可用图 1.3.1 表示:

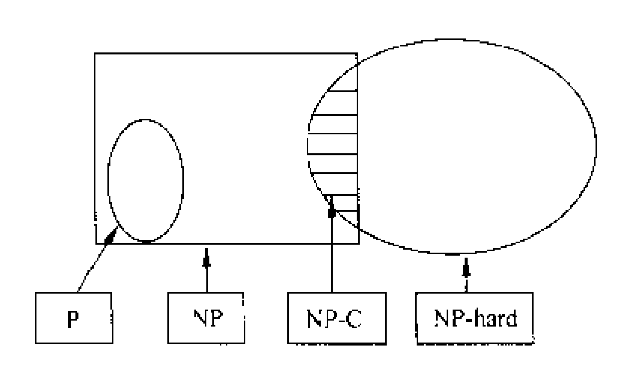


图 1.3.1 四类问题的关系图

NP-C 问题具有重要的实际意义和工程背景,目前已有许多问题被证明为NP-C,如背包问题、装箱问题、Job Shop 和 Flow Shop 问题、集合覆盖问题、旅行商问题等,有兴趣的读者可阅读 Garey 和 Johnson 的著作"Computers and Intractability: A Guide to the Theory of NP-Completeness"。针对这种类型问题的研究主要包括:提出改进算法或启发式算法,与现有算法作比较,分析其 Worst-Case 性能、计算复杂性和收敛性能;将问题或模型简单化,提出基于简化模型的高效方法,分析与原问题解的差距;将实际问题或未研究问题跟已知问题进行对照分析或转换,进而用已有方法解决这些问题。

对于生产调度问题,除少数小规模问题存在 P 算法外,大量调度问题属于 NP-hard问题,至今没有找到可以精确求得最优解的多项式时间算法。鉴于精确求

解方法仅适合于小规模问题，构造性方法优化质量较差且缺少柔性，先进的邻域搜索方法等智能优化算法在生产调度领域受到广泛重视和研究，其中以遗传算法的研究居多，而本书正是介绍基于遗传算法的生产调度问题的优化与设计。在后续章节中，我们将首先介绍遗传算法的理论与实现技术，然后将围绕不同类型的生产调度问题（主要是静态车间调度问题），着重介绍实效性遗传算法的设计。

第2章 遗传算法理论与实现技术

遗传算法(GA)是J. Holland于1975年受生物进化论的启发而提出的。GA的提出一定程度上解决了传统的基于符号处理机制的人工智能方法在知识表示、信息处理和解决组合爆炸等方面所遇到的困难,其自组织、自适应、自学习和群体进化能力使其适合于大规模复杂优化问题。GA是基于"适者生存"的一种高度并行、随机和自适应优化算法,它将问题的求解表示成"染色体"的适者生存过程,通过"染色体"群(population)的一代代不断进化,包括复制(reproduction)、交叉(crossover)和变异(mutation)等操作,最终收敛到"最适应环境"的个体,从而求得问题的最优解或满意解。GA是一种通用的优化算法,其编码技术和遗传操作比较简单,优化不受限制性条件的约束,而其两个最大的显著特点则是隐含并行性和全局解空间搜索。目前,随着计算机技术的发展,GA愈来愈得到人们的重视,并在机器学习、模式识别、图像处理、神经网络、优化控制、组合优化、VLSI设计、遗传学等领域得到了成功应用,尤其在生产调度领域。本章将分别介绍GA的优化流程、操作、算法理论与技术,为后文介绍GA在生产调度中的应用打下基础。

2.1 遗传算法的基本流程

通常,把遗传算法(GA)、进化规划(EP)、进化策略(ES)和遗传编程(GP)统称为进化计算(EC)。其中,GA着重强调基因层次的进化。GA是一类随机优化算法,但它不是简单的随机比较搜索,而是通过对染色体(chromosome)或个体(individual)或串(string)的评价和对染色体及其基因(gene)的作用,有效地利用已有信息来指导搜索有希望改善优化质量的状态。

[**标准遗传算法**]:

步骤1:令$k=0$,随机产生N个初始个体构成初始种群$P(0)$。

步骤2:评价$P(k)$中各个体的适配值(fitness value)。

步骤3:判断算法收敛准则是否满足。若满足则输出搜索结果;否则执行以下步骤。

步骤4:令$m=0$。

步骤5:根据适配值大小以一定方式执行复制操作来从$P(k)$中选取两个个体。

步骤6:若交叉概率$p_c>\xi\in[0,1]$,则对选中个体执行交叉操作来产生两个临时个体;否则将所选中父代个体作为临时个体。

步骤 7：按变异概率 p_m 对临时个体执行变异操作产生两个新个体放入 $P(k+1)$，并令 $m=m+2$。

步骤 8：若 $m<N$，则返回步骤 5；否则令 $k=k+1$，并返回步骤 2。

上述算法中，适配值是对个体进行评价的一种指标，是 GA 进行优化所用的主要信息，它与个体的目标值存在一种对应关系；复制操作（也称选择操作）通常采用比例复制，即复制概率正比于个体的适配值，如此意味着适配值高的个体在下一代中复制自身的概率大，从而可提高种群的平均适配值；交叉操作通过交换两父代个体的部分信息构成后代个体，使得后代继承父代的有效模式，从而有助于产生优良个体；变异操作通过随机改变个体中某些基因而产生新个体，有助于增加种群的多样性，避免早熟收敛。

标准遗传算法的流程图描述，如图 2.1.1 所示。

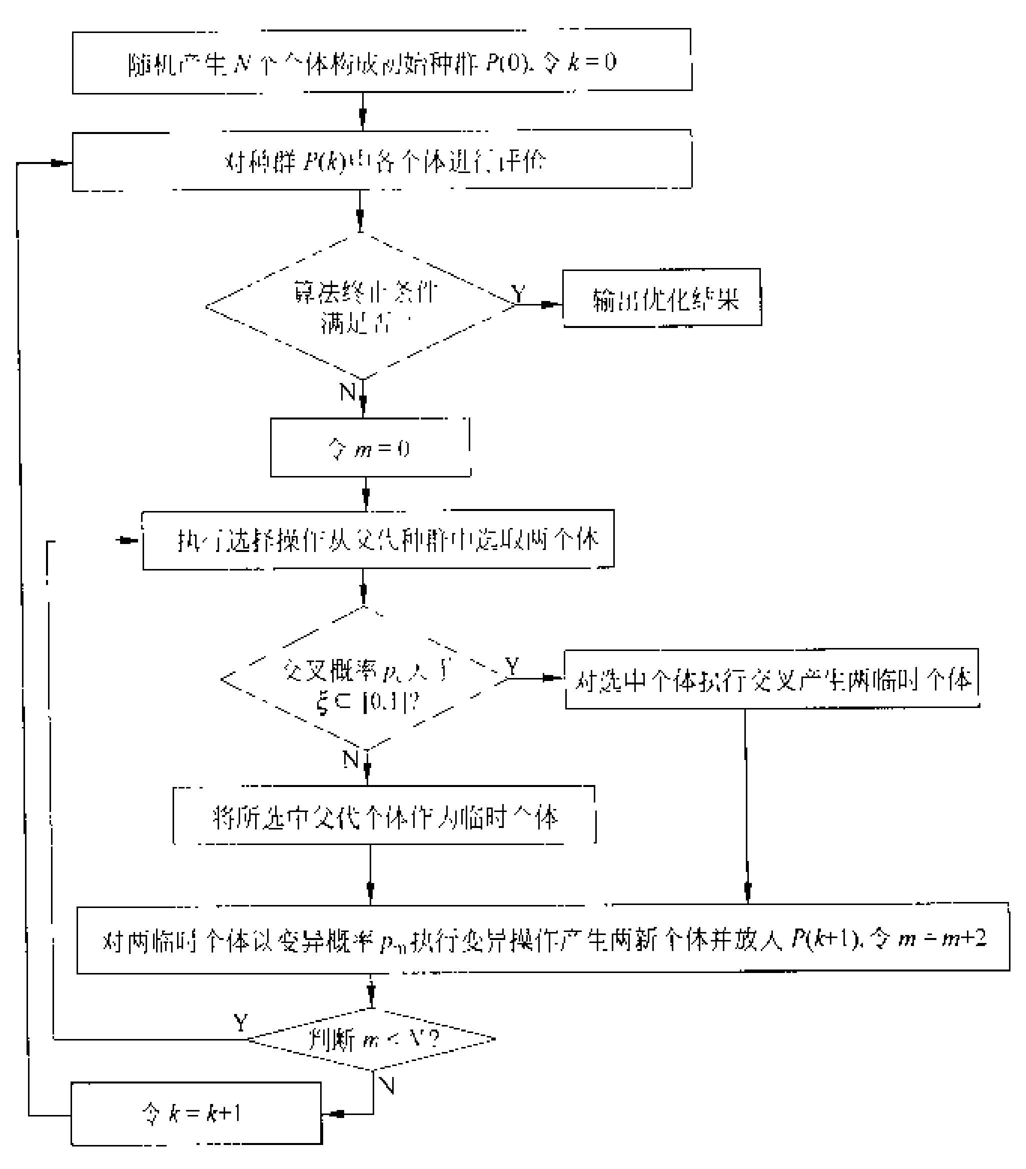


图 2.1.1　标准遗传算法框图

当然迄今遗传算法还有很多变型或改进算法(见 2.6 节算法改进),但其基于生物进化和遗传思想实现优化过程的机制没变。区别于传统优化算法,它具有以下特点:

(1) GA 对问题参数编码成"染色体"后进行进化操作,而不是针对参数本身,这使得 GA 不受函数约束条件的限制,如连续性、可导性等。

(2) GA 的搜索过程是从问题解的一个集合开始的,而不是从单个个体开始的,具有隐含并行搜索特性,从而大大减小了陷入局部极小的可能。

(3) GA 使用的遗传操作均是随机操作,同时 GA 根据个体的适配值信息进行搜索,而无需其他信息,如导数信息等。

(4) GA 具有全局搜索能力,最善于搜索复杂问题和非线性问题。

遗传算法的优越性主要表现在:

(1) 算法进行全空间并行搜索,并将搜索重点集中于性能高的部分,从而能够提高效率且不易陷入局部极小。

(2) 算法具有固有的并行性,通过对种群的遗传处理可处理大量的模式,并且容易并行实现。

2.2 模式定理和隐含并行性

模式定理和隐含并行性是 Holland 为解释基于二进制编码的标准遗传算法(SGA)的功效而建立的,是最早对遗传算法全局收敛性作定性分析的理论基础,它说明了适配值高、长度短、阶数低的图式在后代中至少以指数增长包含该图式的个体数。

基于二进制编码的 SGA 中,个体是以二进制字符串形式表示的,个体的每一位称为基因,令 X_l 为长度为 l 的二进制串的全体。若用 * 表示通配符,即表示该位置的基因可取 1 或 0,则空间 $V^l=\{0,1,*\}^l$ 表示所有模式的集合。譬如 $l=5$ 时,模式 $H=01**1$ 表示集合$\{01001,01011,01101,01111\}$。称出现在模式中取确定值的位置的数目为模式的阶,记为 $o(H)$,如 $o(01**1)=3$。称模式中第一个取确定值的位置与最后一个取确定值的位置之间的距离为模式的对应长度,记为 $\delta(H)$,如 $\delta(01**1)=4$,$\delta(*0***)=0$。

设种群数目为 N,令 $X=(B^l)^N$ 为种群状态空间,$H\in V^l$ 为任一模式,记算法中第 t 代种群为 $P(t)=\{x_1(t),x_2(t),\cdots,x_N(t)\}\in X$,$f:(B^l)^N\to R^1$ 为适配值函数,则称 $f(H,t)=\frac{1}{|H\cap P(t)|}\sum_{x\in H\cap P(t)}f(x)$ 为模式 H 的平均适配值,其中$|H\cap P(t)|$表示 $H\cap P(t)$ 中元素的个数,即 $P(t)$ 中含有 H 中元素的个数,并称 $\bar{f}(t)=\sum_{x\in P(t)}f(x)/N$ 为 $P(t)$的平均适配值。

模式定理 设 SGA 采用比例选择策略，交叉概率和变异概率分别为 p_c 和 p_m，且 p_m 取值较小，模式 H 的定义长度为 $\delta(H)$，阶为 $o(H)$，第 $t+1$ 代种群 $P(t+1)$ 含有 H 中元素个数的期望为 $E[|H\cap P(t+1)|]$，则以下不等式成立：

$$E[|H\cap P(t+1)|]\geqslant |H\cap P(t)|\cdot\frac{f(H,t)}{\bar{f}(t)}\cdot\left[1-p_c\frac{\delta(H)}{l-1}-o(H)p_m\right]$$

证明 由于 SGA 按适配值采用比例选择策略，则 $H\cap P(t)$ 中元素得以选择的期望值为

$$\sum_{x\in H\cap P(t)}\frac{f(x)}{\bar{f}}=|H\cap P(t)|\cdot\frac{f(H,t)}{\bar{f}(t)}$$

由于交叉操作是随机选取 1 到 $l-1$ 中的某一位置并交换两父代的对应子串，则属于模式 H 的个体经交叉后仍属于 H 的概率不小于 $1-p_c\frac{\delta(H)}{l-1}$。而每个个体经变异操作后未被破坏的概率为 $(1-p_m)^{o(H)}$。此外，$P(t)$ 不属于 H 的个体也有可能经交叉、变异后属于 H。因此，$P(t+1)$ 中属于 H 的个体数目的期望值不小于 $|H\cap P(t)|\cdot\frac{f(H,t)}{\bar{f}(t)}\cdot\left[1-p_c\frac{\delta(H)}{l-1}\right]\cdot(1-p_m)^{o(H)}$。

通常 SGA 中 p_m 取值很小，则

$$\begin{aligned}\left[1-p_c\frac{\delta(H)}{l-1}\right]\cdot(1-p_m)^{o(H)}&\approx\left[1-p_c\frac{\delta(H)}{l-1}\right]\cdot[1-o(H)p_m]\\&\geqslant 1-p_c\frac{\delta(H)}{l-1}-o(H)p_m\end{aligned}$$

从而，模式定理得证。

推论 在 SGA 中，阶次低、定义长度短且适配值超过平均适配值的模式在种群中的数目的期望值以指数级递增。

尽管模式定理一定意义上解释了 SGA 的有效性，但还存在以下缺点：

(1) 模式定理仅适用于基于二进制编码的 SGA，对其他编码方式此定理未必成立；

(2) 模式定理仅提供一个期望值的下界，但仍不能说明算法的收敛性；

(3) 模式定理对算法参数的选取不能够提供实用的指导，对算法操作也有依赖性。

隐含并行性定理 设 ϵ 为一小正数，$l_s<\epsilon(l-1)+1$，$N=2^{l_s/2}$，则 SGA 一次处理的存活概率不小于 $1-\epsilon$ 且定义长度不大于 l_s 的模式数为 $O(N^3)$。

此定理说明 SGA 表面上每代仅对 N 个个体作处理，但实际上并行处理了大约 $O(N^3)$ 个模式，并且无需额外的存储，这正是遗传算法具有高效搜索能力的所在，即隐含并行性。

由于模式定理和隐含并行性定理无助于严格解释算法的收敛性，下面我们将利

用马尔可夫链描述遗传算法，并探讨其收敛性。

2.3 遗传算法的马尔可夫链描述及其收敛性

2.3.1 预备知识

定义 2.3.1 称 $n\times n$ 方阵 $\boldsymbol{A}=(a_{ij})$ 为

(1) 非负的(non-negative)，记 $\boldsymbol{A}\geqslant 0$，若 $a_{ij}\geqslant 0, i,j=1,2,\cdots,n$；

(2) 严格正的(positive)，记 $\boldsymbol{A}>0$，若 $a_{ij}>0, i,j=1,2,\cdots,n$；

(3) 正则的(primitive)，若 $\boldsymbol{A}\geqslant 0$，且存在自然数 k，使 $\boldsymbol{A}^k>0$；

(4) 随机的(stochastic)，若 $\boldsymbol{A}\geqslant 0$，$\sum_{j=1}^{n}a_{ij}=1$，$i=1,2,\cdots,n$；

(5) 归约的(reducible)，若 $\boldsymbol{A}\geqslant 0$ 且经过相同的行和列初等变换后可转化为 $\begin{bmatrix}\boldsymbol{C} & \boldsymbol{0}\\ \boldsymbol{R} & \boldsymbol{T}\end{bmatrix}$ 的形式，其中 $\boldsymbol{C}$ 和 $\boldsymbol{T}$ 均为方阵；

(6) 不可约的(non-reducible)，若 $\boldsymbol{A}\geqslant 0$ 且不归约；

(7) 稳定的(stable)，若 $\boldsymbol{A}$ 是随机的且所有行相同；

(8) 列容的(column allowable)，若 $\boldsymbol{A}$ 是随机的且每一列中至少有一个正数。

引理 2.3.1 若 $\boldsymbol{C}$、$\boldsymbol{M}$ 和 $\boldsymbol{S}$ 为随机阵，且 $\boldsymbol{M}>0$，$\boldsymbol{S}$ 为列容的，则 $\boldsymbol{CMS}>0$。

证明 记 $\boldsymbol{A}=\boldsymbol{CM}$，$\boldsymbol{B}=\boldsymbol{AS}$。由于 $\boldsymbol{C}$ 为随机阵，则 $\boldsymbol{C}$ 的每一行中至少存在一个正数。从而 $\forall i,j\in\{1,2,\cdots,n\}$，$a_{ij}=\sum_{k=1}^{n}c_{ik}m_{kj}>0$，即 $\boldsymbol{A}>0$。进而，由于 $\boldsymbol{S}$ 为列容的，则 $\forall i,j\in\{1,2,\cdots,n\}$，$b_{ij}=\sum_{k=1}^{n}a_{ik}s_{kj}>0$。故 $\boldsymbol{CMS}>0$ 得证。

引理 2.3.2 若 $\boldsymbol{P}$ 为正则随机阵，则 $\boldsymbol{P}^k$ 收敛到一个全正稳定随机阵，即

$$\boldsymbol{P}^{\infty}=[1,1,\cdots,1]^{\mathrm{T}}[p_1,p_2,\cdots,p_n]$$

其中 $[p_1,p_2,\cdots,p_n]\boldsymbol{P}=[p_1,p_2,\cdots,p_n]$，$\sum_{j=1}^{n}p_{ij}=1$，$p_j=\lim_{k\to\infty}p_{ij}^{(k)}>0$，即 $[p_1,p_2,\cdots,p_n]^{\mathrm{T}}$ 是 $\boldsymbol{P}^{\mathrm{T}}$ 的特征值为 1 且各分量均为正数的特征向量。

引理 2.3.3 若 $\boldsymbol{P}=\begin{bmatrix}\boldsymbol{C} & \boldsymbol{0}\\ \boldsymbol{R} & \boldsymbol{T}\end{bmatrix}$ 是随机的，其中 C 为 m 维严格正随机方阵，$\boldsymbol{R}\neq 0$，$\boldsymbol{T}\neq 0$，则 $\boldsymbol{P}^{\infty}=\begin{bmatrix}\boldsymbol{C}^{\infty} & \boldsymbol{0}\\ \boldsymbol{R}^{\infty} & \boldsymbol{0}\end{bmatrix}$ 是一个稳定的随机阵，其中 $\boldsymbol{R}^{\infty}=\lim_{k\to\infty}\sum_{i=0}^{k-1}\boldsymbol{T}^i\boldsymbol{R}\boldsymbol{C}^{k-i}$，同时

$$\boldsymbol{P}^{\infty}=[1,1,\cdots,1]^{\mathrm{T}}[p_1,p_2,\cdots,p_n],\quad \sum_{j=1}^{n}p_{ij}=1,\quad p_j=\lim_{k\to\infty}p_{ij}^{(k)}\geqslant 0$$

其中 $p_j>0(1\leqslant j\leqslant m)$，$p_j=0(m+1\leqslant j\leqslant n)$。

2.3.2 标准遗传算法的马尔可夫链描述

在 SGA 中,令所有个体形成的有限空间为 $\boldsymbol{\Omega}$,所有种群对应的整个状态空间为 $\boldsymbol{G}$,其中每一种群为一个状态,一旦染色体长度 l 和种群数目 N 给定且有限,则 $\boldsymbol{\Omega}$ 的维数 $|\boldsymbol{\Omega}|$ 有限,$\boldsymbol{G}$ 的维数 $|\boldsymbol{G}|=|\boldsymbol{\Omega}|^N$ 也有限。由于算法中每一代状态的转移依赖于选择(复制)、交叉和变异操作,且与进化代数无关,因此 SGA 可视为一个有限状态的齐次马尔可夫链。

鉴于算法中三个遗传操作是循环往返执行的,我们按"交叉→变异→选择"顺序来考虑算法中状态的转移。于是,表征马尔可夫链的状态转移矩阵 $\boldsymbol{P}$ 为矩阵 $\boldsymbol{C}$,$\boldsymbol{M}$ 和 $\boldsymbol{S}$ 的乘积,其中 $\boldsymbol{C}$、$\boldsymbol{M}$ 和 $\boldsymbol{S}$ 分别为交叉、变异和选择操作所决定的状态转移矩阵。

1. 交叉操作

交叉操作可视为状态空间上的随机函数,记交叉操作决定的状态转移矩阵为 $\boldsymbol{C}=(c_{ij})_{|G|\times|G|}$,其中 c_{ij} 为从状态 i 经交叉操作转移到状态 j 的概率。由于一个状态通过交叉操作总要转移到状态空间中的另一个状态,因此 $\sum_{j=1}^{|G|}c_{ij}=1$ 显然成立,即 $\boldsymbol{C}$ 为随机矩阵。

2. 变异操作

在 SGA 中,变异操作是独立作用于种群内各个体的每一位基因上,记变异操作决定的状态转移矩阵为 $\boldsymbol{M}=(m_{ij})_{|G|\times|G|}$,其中 m_{ij} 为从状态 i 经变异操作转移到状态 j 的概率。由于染色体的每个基因只有相同的变异概率 $p_m>0$,则

$$m_{ij}=p_m^{H_{ij}}(1-p_m)^{N\cdot l-H_{ij}}>0$$

其中 H_{ij} 为状态 i 与状态 j 之间具有不同基因的位置的总数。例如,状态{(1000),(0111),(1010)}与状态{(1100),(1110),(0101)}之间的 H_{ij} 为 7。因此,$\boldsymbol{M}$ 为严格正的随机矩阵。

3. 选择操作

在 SGA 中,交叉和变异操作后得到的种群经选择操作后又将转移到另一个种群,记选择操作决定的状态转移矩阵为 $\boldsymbol{S}=(s_{ij})_{|G|\times|G|}$。考虑比例选择(轮盘赌)策略,种群中染色体 X_i 被选中的概率为 $f(X_i)\Big/\sum_{k=1}^{N}f(X_k)>0$,则种群 i 经选择操作后保持不变的概率 $s_{ii}\geqslant\prod_{j=1}^{N}\left(f(X_j)\Big/\sum_{k=1}^{N}f(X_k)\right)>0$。因此,转移矩阵 $\boldsymbol{S}$ 是随机且列容的。

通过对上述三个操作的概率描述,我们对 SGA 的马尔可夫链描述有了一个清晰的认识,下面将对算法的收敛性进行阐述。

2.3.3 标准遗传算法的收敛性

本节仅讨论基于二进制编码的 SGA 的收敛性，后文将讨论一般遗传算法的收敛性。

定理 2.3.1 若 SGA 中交叉概率 $p_c \in [0,1]$，变异概率 $p_m \in [0,1]$，并且算法采用比例选择策略，则 SGA 的状态转移矩阵 $\boldsymbol{P}=\boldsymbol{CMS}$ 是严格正的。

证明 利用上节矩阵 $\boldsymbol{C}$，$\boldsymbol{M}$ 和 $\boldsymbol{S}$ 的定义以及引理 2.3.1 即可。

注：根据上述定理可知，SGA 是一个遍历马尔可夫链，从而存在一个与初值无关的极限分布，由此算法的初始种群可以任意初始化，同时任何时刻从一个状态转移到另一个状态的概率不为 0，即马尔可夫链构成了强连通图。然而，尽管初始种群的随机性在理论上不影响极限分布，但由于实际算法中某些环节的近似处理(如终止准则)，算法的搜索结果存在一定的波动性，通常算法要执行多次才能得到较可靠的结果。

定理 2.3.2 SGA 不能够以概率 1 收敛到全局最优解。

证明 令 $P(t)=\{X_1(t),\cdots,X_N(t)\}$ 为算法第 t 代的种群，其最优适配值为

$$Z_t = \max\{f(X_k(t)),k=1,\cdots,N\}$$

设 f^* 为全局最优的适配值。由定理 2.3.1 和引理 2.3.2 知，SGA 以任意状态 i 为极限分布的概率 $\lim\limits_{t\to\infty} p_i^t = p_i^\infty > 0$，因此，

$$\lim_{t\to\infty} P\{Z_t = f^*\} \leqslant 1 - p_i^\infty < 1$$

注：上述定理从概率意义上说明了 SGA 不能够收敛到全局最优，其原因在于算法中最优解的概率遗失。因此，只要在算法中每代保留当前最优解，无论是在选择之前还是在选择之后，算法将最终收敛到全局最优，从而有以下定理。

定理 2.3.3 每代在选择操作后保留最优解的 SGA 以概率 1 收敛到全局最优解。

证明 由于算法采用了保优技术，为方便起见，我们将保留下来的各代的最优个体存放在种群的第一号位置，但它不参与遗传操作。即在第 t 代进化时的种群为 $(X^*(t-1),X_1(t),\cdots,X_N(t))$，其中 $X^*(t-1)$ 表示算法搜索至第 $t-1$ 代所得到的最佳个体。于是，算法对应马尔可夫链的状态空间维数将变为 $|\boldsymbol{\Omega}|^{N+1}$。然后，我们将包含相同的最优解的状态仍按在原状态空间的顺序进行排列，而对包含不同最优解的状态按最优解的适配值从大到小进行排列。由此，新的交叉、变异和选择操作对应的状态转移矩阵可表示为 $\boldsymbol{C}^+=\mathrm{diag}(\boldsymbol{C},\boldsymbol{C},\cdots,\boldsymbol{C})$，$\boldsymbol{M}^+=\mathrm{diag}(\boldsymbol{M},\boldsymbol{M},\cdots,\boldsymbol{M})$ 和 $\boldsymbol{S}^+=\mathrm{diag}(\boldsymbol{S},\boldsymbol{S},\cdots,\boldsymbol{S})$。

在选择操作后，算法要将当前种群中的最优解与前一代保留下来的最优解进行比较，这一操作也可用矩阵 $\boldsymbol{U}=(u_{ij})$ 表示。显然，状态 $Y(t)=[X^*(t-1),X_1(t),\cdots,$

$X_N(t)$]转移到 $Y(t+1)=[X^*(t),X_1(t+1),\cdots,X_N(t+1)]$的概率为

$$\Pr[Y(t+1)\mid Y(t)]=\begin{cases}1 & \text{if } \max(f(X_1(t)),\cdots,f(X_N(t)))=X^*(t)\\ & \text{and } (X_1(t),\cdots,X_N(t))=(X_1(t+1),\cdots,X_N(t+1))\\ 0 & \text{else}\end{cases}$$

因此,矩阵 $\boldsymbol{U}$ 每一行均有且只有一个元素为 1,而其他元素为 0。同时,考虑到状态的排列顺序,可知矩阵 $\boldsymbol{U}$ 为下三角矩阵。记

$$\boldsymbol{U}=\begin{bmatrix}\boldsymbol{U}_{11} & & & \\ \boldsymbol{U}_{21} & \boldsymbol{U}_{22} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{U}_{|\boldsymbol{\Omega}|1} & \boldsymbol{U}_{|\boldsymbol{\Omega}|2} & \cdots & \boldsymbol{U}_{|\boldsymbol{\Omega}||\boldsymbol{\Omega}|}\end{bmatrix}$$

其中 $\boldsymbol{U}_{ij}$ 为 $|\boldsymbol{\Omega}|^N\times|\boldsymbol{\Omega}|^N$ 阶矩阵,且 $\boldsymbol{U}_{11}$ 为单位矩阵。于是,马尔可夫链的一步转移矩阵为

$$\boldsymbol{P}^+=\boldsymbol{C}^+\boldsymbol{M}^+\boldsymbol{S}^+\boldsymbol{U}=\mathrm{diag}(\boldsymbol{CMS},\boldsymbol{CMS},\cdots,\boldsymbol{CMS})$$

$$=\begin{bmatrix}\boldsymbol{CMSU}_{11} & & & \\ \boldsymbol{CMSU}_{21} & \boldsymbol{CMSU}_{22} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{CMSU}_{|\boldsymbol{\Omega}|1} & \boldsymbol{CMSU}_{|\boldsymbol{\Omega}|2} & \cdots & \boldsymbol{CMSU}_{|\boldsymbol{\Omega}||\boldsymbol{\Omega}|}\end{bmatrix}$$

显然,$\boldsymbol{P}^+$ 为可约的随机矩阵,且 $\boldsymbol{CMSU}_{11}$ 严格正。进而,考虑到 $\boldsymbol{P}^+$ 中第一位状态由好到坏的排列顺序可知

$$\boldsymbol{R}=\begin{bmatrix}\boldsymbol{CMSU}_{21}\\ \vdots \\ \boldsymbol{CMSU}_{|\boldsymbol{\Omega}|1}\end{bmatrix}\neq 0,\quad \boldsymbol{T}=\begin{bmatrix}\boldsymbol{CMSU}_{22} & & \\ \vdots & \ddots & \\ \boldsymbol{CMSU}_{|\boldsymbol{\Omega}|2} & \cdots & \boldsymbol{CMSU}_{|\boldsymbol{\Omega}||\boldsymbol{\Omega}|}\end{bmatrix}\neq 0$$

由引理 2.3.3 可知,不包含最优个体的状态在马尔可夫链的极限分布中的概率为 0,包含最优个体的状态在马尔可夫链的极限分布中的概率和为 1。从而,算法将以概率 1 收敛到包含全局最优个体的状态,也即算法能够以概率 1 搜索到全局最优解。

定理 2.3.4 每代在选择操作前保留最优解的 SGA 以概率 1 收敛到全局最优解。

证明过程类似于定理 2.3.3,在此不再赘述。

2.3.4 标准遗传算法的收敛速度估计

此节简单介绍带有保优操作的 SGA 的收敛速度估计。

设带有保优操作的遗传算法对应于有限状态维马尔可夫链$\{Z_k\}$,行向量 $\boldsymbol{\pi}$ 为给定每一状态的概率,$\boldsymbol{P}$ 为一步转移矩阵,则经 n 次转移后,各状态的概率为 $\boldsymbol{\pi P}^n$。若种群中全部个体相同且均为最优,则称此状态为吸收态。显然,马尔可夫链$\{Z_k\}$中

有如下三种性质的状态：

(1) 吸收态。

(2) 可一步转移到吸收态的非吸收态。

(3) 经 $\boldsymbol{P}$ 可一步转移到另一状态，而此状态转移到吸收态的概率为零。因此，一步转移矩阵 $\boldsymbol{P}$ 可分解为

$$\boldsymbol{P}=\begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{0}\\ \boldsymbol{R} & \boldsymbol{Q}\end{bmatrix}$$

其中，$\boldsymbol{I}_k$ 为描述吸收态的 k 阶单位矩阵，$\boldsymbol{R}$ 为描述能转移到吸收态的状态的 $(|G|-k)\times k$ 阶矩阵，$\boldsymbol{Q}$ 为描述其他状态的 $(|G|-k)\times(|G|-k)$ 阶矩阵。

进而可得

$$\boldsymbol{P}^n=\begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{0}\\ \boldsymbol{M}_n\boldsymbol{R} & \boldsymbol{Q}^n\end{bmatrix}$$

其中 $\boldsymbol{M}_n=\boldsymbol{I}+\boldsymbol{Q}+\boldsymbol{Q}^2+\cdots+\boldsymbol{Q}^{n-1}$，$\|Q\|<1$。于是有

$$\boldsymbol{P}^\infty=\lim_{n\to\infty}\boldsymbol{P}^n=\begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{0}\\ (\boldsymbol{I}-\boldsymbol{Q})^{-1}\boldsymbol{R} & \boldsymbol{0}\end{bmatrix}$$

由于保优 SGA 算法的全局收敛性，最优分布 z^* 存在，且 $z^*=z(0)\boldsymbol{P}^\infty$，$z(0)=[z_1(0),\cdots,z_k(0),z_{k+1}(0),\cdots,z_{|\Omega|}(0)]=[\bar{z}_1(0),\bar{z}_2(0)]$。以下给出 $z(i)=z(0)\boldsymbol{P}^i$ 收敛到 z^* 的速度估计。

定理 2.3.5 设 $\|Q\|=\lambda<1$，则概率分布 $z(i)$ 收敛到 z^* 的收敛速度估计为 $\|z(i)-z^*\|\leqslant O(\lambda^i)$。

证明

$$\|z(i)-z^*\|=\|z(0)\boldsymbol{P}^i-z(0)\boldsymbol{P}^\infty\|$$

$$=\left\|z(0)\cdot\left(\begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{0}\\ \boldsymbol{M}_i\boldsymbol{R} & \boldsymbol{Q}^i\end{bmatrix}-\begin{bmatrix}\boldsymbol{I}_k & \boldsymbol{0}\\ (\boldsymbol{I}-\boldsymbol{Q})^{-1}\boldsymbol{R} & \boldsymbol{0}\end{bmatrix}\right)\right\|$$

$$=\left\|z(0)\cdot\begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ (\boldsymbol{M}_i-(\boldsymbol{I}-\boldsymbol{Q})^{-1})\boldsymbol{R} & \boldsymbol{Q}^i\end{bmatrix}\right\|$$

$$=\|\bar{z}_2(0)\cdot[(\boldsymbol{M}_i-(\boldsymbol{I}-\boldsymbol{Q})^{-1})\boldsymbol{R}\quad \boldsymbol{Q}^i]\|$$

$$\leqslant\|\bar{z}_2(0)\|\cdot(\|\boldsymbol{M}_i-(\boldsymbol{I}-\boldsymbol{Q})^{-1}\|\cdot\|\boldsymbol{R}\|+\|\boldsymbol{Q}^i\|)$$

$$=\|\bar{z}_2(0)\|\cdot(\|(\boldsymbol{I}-\boldsymbol{Q})^{-1}-(\boldsymbol{I}-\boldsymbol{Q})^{-1}\boldsymbol{Q}^i-(\boldsymbol{I}-\boldsymbol{Q})^{-1}\|\cdot\|\boldsymbol{R}\|+\|\boldsymbol{Q}^i\|)$$

$$=\|\bar{z}_2(0)\|\cdot(\|-(\boldsymbol{I}-\boldsymbol{Q})^{-1}\|\cdot\|\boldsymbol{R}\|+1)\cdot\|\boldsymbol{Q}^i\|$$

$$\leqslant\|\bar{z}_2(0)\|\cdot\left(\frac{\|\boldsymbol{R}\|}{1-\|\boldsymbol{Q}\|}+1\right)\cdot\|\boldsymbol{Q}\|^i=O(\lambda^i)$$

迄今为止，遗传算法的理论研究仍主要针对 SGA 模型。高级的遗传算法由于其本身的多样性，理论方面的研究相当分散，尚未取得引人注目的结果。下一节将简单

介绍一般可测状态空间和有限离散状态空间上GA的收敛性和收敛速度，其结论可覆盖不同编码机制和遗传操作下的遗传算法，结论更具一般性。同时，大部分结论都是通过计算机数值仿真来说明的，数学上严格完整且令人信服的解释仍需努力探索。GA的收敛性研究，尤其是如何提高算法的收敛速度和鲁棒性，仍是具有重要研究价值的课题。

2.4 一般可测状态空间上遗传算法的收敛性

上节基于有限状态马尔可夫链理论和特征值分析讨论了有限状态空间上SGA的收敛性和收敛速度估计，本节讨论一般可测状态空间和有限离散状态空间上遗传算法的收敛性及其收敛速度估计(He, et al, 1999)。

2.4.1 问题描述

考虑如下一类优化问题：

$$\max\{f(x):x\in X\} \tag{2-4-1}$$

其中，x为优化状态，X为可测的有界紧致空间，$f(x)$为有界适配值函数。

假定问题的最优解集$S_{opt}=\{x;f(x)=f_{max}\}$非空。令$\phi(\cdot)$为空间$X$的一个测度函数，通常$S_{opt}$仅包含有限解，因此$\phi(S_{opt})=0$。为了分析方便，如果$\phi(S_{opt})=0$，则用一个具有正测度的扩大集$S_{opt-\epsilon}=\{x;|f(x)-f_{max}|\leqslant\epsilon\}$来代替$S_{opt}$，其中$\epsilon$为小正实数。因此，基于上述考虑，在此假设$\phi(S_{opt})>0$。

2.4.2 算法及其马尔可夫链描述

遗传算法的框架同SGA，为方便起见，按变异→交叉→选择的顺序进行遗传操作。

定义 2.4.1 称X为个体空间，称X的每个点x为一个个体，称乘积空间$X^N=X\times X\times\cdots\times X$为种群空间，称$X^N$中的每一个$\boldsymbol{x}=\{x_1,\cdots,x_N\}$为包含$N$个个体的一个种群，定义种群$\boldsymbol{x}$的适配值为

$$f(\boldsymbol{x})=\max\{f(x_i):x_i\in\boldsymbol{x}\}$$

定义全局最优解集为

$$S_{opt}^N=\{\boldsymbol{x};\exists x_i\in\boldsymbol{x},x_i\in S_{opt}\}$$

假设$\xi(t)$为t时刻的种群状态，$\xi_m(t)$、$\xi_c(t)$和$\xi_s(t)$分别为变异后、交叉后和选择后的种群状态，则$\xi(t)$、$\xi_m(t)$，$\xi_c(t)$和$\xi_s(t)$均为X^N上的随机变量，且$\xi(t+1)=\xi_s(t)$。各变量的演化顺序为$\xi(t)\to\xi_m(t)\to\xi_c(t)\to\xi_s(t)\to\xi(t+1)$。同时，由于各遗传操作不随时间变化，因此相应的转移概率函数是时不变的。

令变异、交叉和选择操作引起的转移概率函数分别为

$$P_m(\boldsymbol{x},A) = P_m(\xi_m(t) \in A \mid \xi(t) = \boldsymbol{x}) \tag{2-4-2}$$

$$P_c(\boldsymbol{x},A) = P_c(\xi_c(t) \in A \mid \xi_m(t) = \boldsymbol{x}) \tag{2-4-3}$$

$$P_s(\boldsymbol{x},A) = P_s(\xi_s(t) \in A \mid \xi_c(t) = \boldsymbol{x}) \tag{2-4-4}$$

其中 $A \subseteq X^N$。

从而，遗传算法可用状态空间 X^N 上的平稳马尔可夫链 $\{\xi(t); t \in Z^+\}$ 来描述，其转移概率由 K-C 方程(Kolmogorov-Chapman Equation)给出：

$$P(\boldsymbol{x},A) = \int_y \int_z P_m(\boldsymbol{x},\mathrm{d}\boldsymbol{y}) P_c(\boldsymbol{y},\mathrm{d}\boldsymbol{z}) P_s(\boldsymbol{z},A) \tag{2-4-5}$$

2.4.3 收敛性分析和收敛速度估计

在此，我们并不讨论遗传操作的具体形式，仅强调它们使算法实现全局收敛的性质。直观上而言，算法要实现全局收敛，首先要求任意初始种群经有限步转移能够到达全局最优解，其次算法必须有保优操作来防止遗失最优解。因此，我们假设各遗传操作满足如下性质。

性质 2.4.1 对任意时刻 t，若 $\xi(t) \in S_{opt}^N$，则 $\xi_m(t+1) \in S_{opt}^N$；若 $\xi_m(t) \in S_{opt}^N$，则 $\xi_c(t+1) \in S_{opt}^N$；若 $\xi_c(t) \in S_{opt}^N$，则 $\xi_s(t+1) \in S_{opt}^N$。

注：性质 2.4.1 意味着各遗传操作具有保优性。

性质 2.4.2 存在某一时刻 t_0，对于任意初始种群 $\xi(1) = \boldsymbol{x}$，任意可测子集 $A \in S_{opt}^N$ 和某一 $\delta(\boldsymbol{x}) > 0$，转移概率函数满足 $P(1,t_0;\boldsymbol{x},A) \geqslant \delta(\boldsymbol{x})\phi(A)$。

注：性质 2.4.2 意味着对任意初始种群，最优集在有限步数内可达。

假设 μ_1, μ_2 为 X^N 上的两个概率测度，定义它们的整体偏差距离(total variation distance)为

$$\| \mu_1 - \mu_2 \| = \sup_{A \subseteq X^N} | \mu_1(A) - \mu_2(A) |$$

其中 A 为 X^N 上的任意可测子集。假设 π 为 X^N 上的一个概率分布，它是以转移函数 $P(x,A)$ 表征的马尔可夫链 $\{\xi(t); t \in Z^+\}$ 的不变量，当且仅当对 X^N 上的所有可测集 A 有

$$\pi(A) = \int_{X^N} P(\boldsymbol{x},A)\pi(\mathrm{d}\boldsymbol{x})$$

定义 2.4.2 对于以转移函数 $P(\boldsymbol{x},A)$ 表征的马尔可夫链 $\{\xi(t); t \in Z^+\}$，记 μ_1 为初始种群对应 $\xi(1)$ 的概率分布，μ_t 为 $\xi(t)$ 的概率分布，若存在 P 的不变概率分布 π 使得 $\lim_{t \to \infty} \| \mu_t - \pi \| = 0$，则称 μ_t 收敛到 π。进而，若存在子集 $A \subseteq X^N$，使得 $\pi(A) = 1$，则称 μ_t 全局收敛到 A。

由此，给出一般可测状态空间上遗传算法的收敛定理和收敛速度估计。

定理 2.4.1 设 X^N 为有界且紧致的可测空间，若以式(2-4-5)表征的马尔可夫

链$\{\xi(t);t\in Z^+\}$满足性质 2.4.1 和性质 2.4.2，则对于任意初始种群 $\xi(1)=\boldsymbol{x}$，马尔可夫链全局收敛到最优集，即存在不变概率测度 π 使得

$$\pi(S_{\mathrm{opt}}^N)=1 \quad 且 \quad \lim_{t\to\infty}\|\mu_t-\pi\|=0$$

同时，存在 $t_0>0$ 和正实数 $\delta>0$，使得

$$\|\mu_t-\pi\|\leqslant(1-\delta)^{\lfloor t/t_0\rfloor-1}$$

其中 μ_t 为 $\xi(t)$ 的概率分布，$\lfloor t/t_0\rfloor$ 表示取不超过 t/t_0 的最大整数。

在证明此定理之前，先介绍 Minorization 条件、Doeblin 条件和若干引理。

定义 2.4.3 称 X^N 上的转移函数 $P(\boldsymbol{x},A)$ 在子集 $R\subseteq X^N$ 上满足 Minorization 条件，若存在 X^N 上的一个概率测度 ψ，正整数 t_0 和正实数 $\delta>0$，使得对所有 $\boldsymbol{x}\in R$ 和 R 上的所有可测集 A，$P(1,t_0;\boldsymbol{x},A)\geqslant\delta\phi(A)$。

定义 2.4.4 定义 2.4.3 中，若 $R=X^N$，则称 Minorization 条件为 Doeblin 条件。

引理 2.4.1 设 X^N 为有界且紧致的可测空间，若以式(2-4-5)表征的马尔可夫链$\{\xi(t);t\in Z^+\}$满足性质 2.4.1 和性质 2.4.2，则它满足 Doeblin 条件。

证明 定义空间 X^N 上的概率测度 ψ 如下：

对所有可测子集 A，$\psi(A)=\phi(A\cap S_{\mathrm{opt}}^N)/\phi(S_{\mathrm{opt}}^N)$。

对于初始种群 $\xi(1)=\boldsymbol{x}$ 和任意可测子集 $A\subseteq X^N$，考虑如下两种情况：

(1) 若 $\phi(A\cap S_{\mathrm{opt}}^N)=0$，则 $\psi(A)=0$，从而 $P(1,t_0;\boldsymbol{x},A)\geqslant\psi(A)=0$ 一直成立。

(2) 若 $\phi(A\cap S_{\mathrm{opt}}^N)>0$，则 $\psi(A)>0$，由性质 2.4.2 知

$$P(1,t_0;\boldsymbol{x},A)\geqslant P(1,t_0;\boldsymbol{x},A\cap S_{\mathrm{opt}}^N)\geqslant\delta(\boldsymbol{x})\phi(A\cap S_{\mathrm{opt}}^N)=\delta(\boldsymbol{x})\phi(S_{\mathrm{opt}}^N)\psi(A)$$

由于 X^N 为有界且紧致的可测空间，因此

$$\delta=\inf\{\delta(\boldsymbol{x})\phi(S_{\mathrm{opt}}^N);\boldsymbol{x}\in X^N\}>0$$

联合考虑上述两种情况得，对于时刻 t_0，正实数 $\delta>0$ 和初始种群 $\boldsymbol{x}$，$P(1,t_0;\boldsymbol{x},A)\geqslant\delta\psi(A)$ 成立，即马尔可夫链满足 Doeblin 条件。

注：t_0 可视为由任意初始种群到 S_{opt}^N 的首达时间，δ 可认为是相应的首达概率。

引理 2.4.2 若以式(2-4-5)表征的马尔可夫链$\{\xi(t);t\in Z^+\}$满足性质 2.4.1 和性质 2.4.2，则对 P 的任意不变概率分布 π，有 $\pi(S_{\mathrm{opt}}^N)=1$。

证明 假设 $\pi(S_{\mathrm{opt}}^N)=1$ 不成立，即 $\pi(X^N-S_{\mathrm{opt}}^N)>0$。

由性质 2.4.1，对于任意种群 $\boldsymbol{x}\in S_{\mathrm{opt}}^N$ 和集合 $X^N-S_{\mathrm{opt}}^N$，有

$$P(\boldsymbol{x},X^N-S_{\mathrm{opt}}^N)=0$$

既然 π 为 P 的任意不变概率分布，则对任意可测集 A，有

$$\pi(A)=\int_{X^N}P(1,t_0;\boldsymbol{x},A)\pi\mathrm{d}\boldsymbol{x}$$

为简单起见，记

$$\pi(A)=\int_{X^N}P^{t_0}(\boldsymbol{x},A)\pi\mathrm{d}\boldsymbol{x}$$

取 $A=X^N-S^N_{opt}$，结合 $P(\boldsymbol{x},X^N-S^N_{opt})=0$，得

$$\pi(X^N-S^N_{opt})=\int_{X^N}P^{t_0}(\boldsymbol{x},X^N-S^N_{opt})\pi\mathrm{d}\boldsymbol{x}$$
$$=\int_{X^N-S^N_{opt}}P^{t_0}(\boldsymbol{x},X^N-S^N_{opt})\pi\mathrm{d}\boldsymbol{x}$$

由性质 2.4.2 知

$$P^{t_0}(\boldsymbol{x},S^N_{opt})>0$$

即

$$P^{t_0}(\boldsymbol{x},X^N-S^N_{opt})<1$$

从而得

$$0<\pi(X^N-S^N_{opt})-\int_{X^N-S^N_{opt}}P^{t_0}(\boldsymbol{x},X^N-S^N_{opt})\pi\mathrm{d}\boldsymbol{x}$$
$$<\int_{X^N-S^N_{opt}}\pi\mathrm{d}\boldsymbol{x}-\pi(X^N-S^N_{opt})$$

这显然是矛盾的，因此 $\pi(S^N_{opt})=1$ 是成立的。

引理 2.4.3 对于可测空间 X^N 上以转移函数 $P(\boldsymbol{x},A)$ 表征的马尔可夫链 $\{\xi(t);t\in Z^+\}$，若它满足 Doeblin 条件，即存在概率测度 ψ、正整数 t_0 和正实数 $\delta>0$，使得

$$P(1,t_0;\boldsymbol{x},A)\geqslant\delta\phi(A)$$

则存在惟一不变量 π，使得

$$\|\mu_t-\pi\|\leqslant(1-\delta)^{[t/t_0]-1}$$

由引理 2.4.1 和引理 2.4.3 知，以马尔可夫链 $\{\xi(t);t\in Z^+\}$ 表征的遗传算法的收敛速度可估计为

$$\|\mu_t-\pi\|\leqslant(1-\delta)^{[t/t_0]-1}$$

同时由引理 2.4.2 知 $\pi(S^N_{opt})=1$，即遗传算法全局收敛到 S^N_{opt}。从而，定理 2.4.1 成立。

由定理 2.4.1 知，收敛速度由首达概率 δ 和首达时间 t_0 控制，增加 δ 或降低 t_0 是提高算法收敛速度的途径。

注：给定任意精度 $\varepsilon>0$，如果希望 $\|\mu_t-\pi\|\leqslant\varepsilon$，仅要求 $(1-\delta)^{[t/t_0]-1}\leqslant\varepsilon$，这就意味着当 $t\geqslant t_0[1+\ln\varepsilon/\ln(1-\delta)]$ 时，遗传算法将达到要求精度。

2.4.4 有限离散状态空间上 GA 的收敛性和收敛速度

对于有限离散状态空间上的 GA，即式(2-4-1)中 X 为有限离散状态空间，则该 GA 对应状态空间 X^N 上的非平稳马尔可夫链 $\{\xi(t);t\in Z^+\}$，其状态转移矩阵如下：

$$\boldsymbol{P}(t;\boldsymbol{x},\boldsymbol{y})=\boldsymbol{P}(\xi(t+1)=\boldsymbol{y}\mid\xi(t)=\boldsymbol{x})$$
$$=\sum_{\boldsymbol{u}\in X^N}\sum_{\boldsymbol{v}\in X^N}\boldsymbol{P}_m(t;\boldsymbol{x},\boldsymbol{u})\boldsymbol{P}_c(t;\boldsymbol{u},\boldsymbol{v})\boldsymbol{P}_s(t;\boldsymbol{v},\boldsymbol{y})\qquad(2\text{-}4\text{-}6)$$

其中

$$\boldsymbol{P}_{m}(t;\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{P}_{m}(\xi_{m}(t) = \boldsymbol{y} \mid \xi(t) = \boldsymbol{x})$$

$$\boldsymbol{P}_{c}(t;\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{P}_{c}(\xi_{c}(t) = \boldsymbol{y} \mid \xi_{m}(t) = \boldsymbol{x})$$

$$\boldsymbol{P}_{s}(t;\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{P}_{s}(\xi_{s}(t) = \boldsymbol{y} \mid \xi_{c}(t) = \boldsymbol{x})$$

分别表征第 t 代进化中变异、交叉和选择操作对应的状态转移。同时，式(2-4-6)的矩阵形式记为

$$\boldsymbol{P}(t) = \boldsymbol{P}_{m}(t) \cdot \boldsymbol{P}_{c}(t) \cdot \boldsymbol{P}_{s}(t)$$

类似一般可测空间上 GA 收敛性的讨论，在此也假设有限离散状态空间上 GA 的遗传操作具有如下性质。

性质 2.4.3 对任意时刻 t，若 $\xi(t) \in S_{\text{opt}}^{N}$，则 $\xi_{m}(t+1) \in S_{\text{opt}}^{N}$；若 $\xi_{m}(t) \in S_{\text{opt}}^{N}$，则 $\xi_{c}(t+1) \in S_{\text{opt}}^{N}$；若 $\xi_{c}(t) \in S_{\text{opt}}^{N}$，则 $\xi_{s}(t+1) \in S_{\text{opt}}^{N}$。

性质 2.4.4 对于矩阵序列$\{\boldsymbol{P}_{m}(t);t=1,2,\cdots\}$，存在矩阵 $\boldsymbol{P}_{m}$ 使得$\lim\limits_{t\to\infty} \| \boldsymbol{P}_{m}(t) - \boldsymbol{P}_{m} \| = 0$ 且 $\sum\limits_{t=1}^{\infty} \sup\limits_{m>0} \| \boldsymbol{P}_{m}(m+t) - \boldsymbol{P}_{m} \|$ 收敛；对于矩阵序列$\{\boldsymbol{P}_{c}(t);t=1,2,\cdots\}$，存在矩阵 $\boldsymbol{P}_{c}$ 使得$\lim\limits_{t\to\infty} \| \boldsymbol{P}_{c}(t) - \boldsymbol{P}_{c} \| = 0$ 且 $\sum\limits_{t=1}^{\infty} \sup\limits_{m>0} \| \boldsymbol{P}_{c}(m+t) - \boldsymbol{P}_{c} \|$ 收敛；对于矩阵序列$\{\boldsymbol{P}_{s}(t);t=1,2,\cdots\}$，存在矩阵 $\boldsymbol{P}_{s}$ 使得$\lim\limits_{t\to\infty} \| \boldsymbol{P}_{s}(t) - \boldsymbol{P}_{s} \| = 0$ 且 $\sum\limits_{t=1}^{\infty} \sup\limits_{m>0} \| \boldsymbol{P}_{s}(m+t) - \boldsymbol{P}_{s} \|$ 收敛。其中，范数 $\| \cdot \|$ 为采用 $\| \cdot \|_{\infty}$。

性质 2.4.5 存在某一时刻 t_{0}，对于任意初始种群 $\boldsymbol{x}$ 和 $\boldsymbol{y} \in S_{\text{opt}}^{N}$，存在某一 $\delta(\boldsymbol{x},\boldsymbol{y})>0$对任意时刻 m 满足

$$\boldsymbol{P}(m,m+t_{0};\boldsymbol{x},\boldsymbol{y}) \geqslant \delta(\boldsymbol{x},\boldsymbol{y})$$

定理 2.4.2 设 X^{N} 为有限离散状态空间，若以式(2-4-6)表征的 GA 对应的马尔可夫链$\{\xi(t);t\in Z^{+}\}$满足性质 2.4.3、性质 2.4.4 和性质 2.4.5，则对于任意初始种群 $\boldsymbol{x}$，存在转移矩阵 $\boldsymbol{P}=\boldsymbol{P}_{m} \cdot \boldsymbol{P}_{c} \cdot \boldsymbol{P}_{s}$ 的一个不变概率分布 π，使得 $\pi(S_{\text{opt}}^{N})=1$，且对于任意时刻 $t>t_{0}$ 和 $m=1,2,\cdots$，满足

$$\| \mu_{m+t} - \pi \| \leqslant \inf_{t_{0}<l<t} \left\{ (1-\delta)^{[l/t_{0}]-1} + \sum_{k=l}^{\infty} a_{k}/\delta \right\}$$

其中 μ_{m+t} 为 $\xi(m+t)$ 的概率分布；$[t/t_{0}]$ 表示取不超过 t/t_{0} 的最大整数；$a_{t} = \sup\limits_{m'>0}\sup\limits_{m''>0} \| \boldsymbol{P}(m'+t) - \boldsymbol{P}(m''+t) \|$；$\delta=\min\{\delta(\boldsymbol{x},\boldsymbol{y});\boldsymbol{x}\in X^{N},\boldsymbol{y}\in S_{\text{opt}}^{N}\}>0$。

注：该定理说明了有限离散状态空间上 GA 的收敛性和收敛速度，若马尔可夫链为平稳，即 $\boldsymbol{P}(m'+t)=\boldsymbol{P}(m''+t)$，则 $a_{t}=0$，收敛速度简化为

$$\| \mu_{m+t} - \pi \| \leqslant \inf_{t_{0}<l<t} (1-\delta)^{[l/t_{0}]-1}$$

总之，遗传算法收敛的条件主要可归纳为遗传操作的保优性和最优种群的可达性。

2.5 遗传算法参数与操作的设计

通常，遗传算法的设计是按以下步骤进行的：

(1) 确定问题的编码方案。由于GA通常不直接作用于问题的解空间，而是利用解的某种编码表示来进行进化，因此选择合理的编码机制对算法质量和效率有很大影响。

(2) 确定适配值函数。由于GA通常基于适配值进行遗传操作，因此合理的适配值能够将各个体的优劣程度得以体现，并适应算法的进化过程。

(3) 算法参数的选取。通常包括种群数目、交叉概率、变异概率、进化代数等。

(4) 遗传算子的设计。通常包括初始化、选择、交叉、变异和替换操作等。

(5) 确定算法的终止条件。终止准则应根据所求解问题的性质，在优化质量和效率方面作合理均衡或侧重。

下面对关键参数与操作的设计作简单介绍。

2.5.1 编码

编码就是将问题的解用一种码来表示，从而将问题的状态空间与GA的码空间相对应，这很大程度上依赖于问题的性质，并将影响遗传操作的设计。由于GA的优化过程不是直接作用于问题参数本身，而是在一定编码机制对应的码空间上进行的，因此编码的选择是影响算法性能与效率的重要因素。

针对函数优化的编码技术主要有二进制编码、十进制编码、实数编码等。

二进制编码将问题的解用一个二进制0-1字符串来表示。譬如采用长度为l的二进制串S来表示$[a,b]$区间内的变量x，则

$$x = a + \frac{[S]_2}{2^l - 1} \cdot (b - a)$$

十进制编码将问题的解用一个十进制串来表示。

显然，不同的码长和码制，对问题求解的精度与效率有很大影响，而且算法也将付出较大的存储量和相应的转换运算。类似地还有Gray编码和串长可变的动态编码。

实数编码将问题的解用一个实数来表示。它解决了二进制和十进制编码对算法精度和存储量的影响，同时便于优化中引入问题的相关信息，譬如梯度信息。实数编码直接在解的表现型上进行遗传操作，目前在高维复杂优化问题中得到广泛应用，并取得了较好效果。

鉴于组合优化问题本身的性质，其编码方式需要特殊设计。常用的组合优化编码技术包括置换排列、0-1矩阵编码等，后文将针对不同类型的调度问题进行阐述。

2.5.2 适配值函数

适配值函数用于对个体进行评价，也是优化过程发展的依据。

对于简单的最小化问题，通常可以直接将目标函数变换成适配值函数（类似于下文的尺度窗口功能），譬如将个体 X 的适配值 $f(X)$ 定义为 $M-c(X)$ 或 $e^{ac(X)}$，其中 M 为一足够大正数，$c(X)$ 为个体的目标值，$a>0$。对于复杂优化问题，往往需要构造合适的评价函数，使其适应 GA 进行优化。

由于适配值度量意义下的个体差异与目标函数值度量意义下的个体差异有所不同，因此若适配值函数设计不当，将难以体现个体的差异，选择操作的作用就很难体现出来，从而造成早熟收敛等缺点。对适配值进行调节是常用的改进方法，如线性变换和指数变换等，即通过某种变换改变原适配值间的比例关系。此外，目前已开发了许多基于序的选择操作，它基于目标值的好坏分配选择概率，无需设计适配值函数，一定程度上缓解了适配值函数定义的难度。

值得一提的是，GA 的优化过程是一种搜索–评价过程，如果评价过程（即适配值或目标值的计算）占有时间过多，则势必影响算法的整体优化性能。因此，对于评价过程复杂的问题，如仿真优化问题，如何提出有效的评价机制来加速或简化评价过程，将有利于提高 GA 的优化效率（王凌等，2002）。

2.5.3 算法参数

Grefenstette(1986)对求解函数优化的标准 GA 定义了 6 个关键参数，包括种群数目（population size，记作 N）、交叉概率（crossover rate，记作 C）、变异概率（mutation rate，记作 M）、代沟（generation gap，记作 G）、尺度窗口（scaling window，记作 W）和选择策略（selection strategy，记作 S）。因此，当 GA 框架确定后，一种 GA 实现即对应一种(N,C,M,G,W,S)参数组合。目前，如何确定 GA 的最优参数仍旧是一个公开的问题，也是研究 GA 的重要课题之一。除了基于大量实验或经验结果确定合适参数外（即所谓的参数调整（parameter tuning）），许多学者设计自适应参数来动态调整 GA 的搜索行为和能力（即所谓的参数控制（parameter control）），而理论性的结论至今仍旧很少。建议读者参阅有关文献（Eiben, et al, 1999）。

1. 种群数目 N

种群数目是影响算法最终优化性能和效率的因素之一。通常，种群太小时，不能提供足够的采样点，以致算法性能很差，甚至得不到问题的可行解；种群太大时，尽管可增加优化信息以阻止早熟收敛的发生，但无疑会增加计算量，从而使收敛时间太长。当然，在优化过程中种群数目是允许变化的，也可以将较大规模的种群分解成若干子种群进行进化。

2. 交叉概率 C

交叉概率用于控制交叉操作的频率。标准 GA 中每一代有 $N\times C$ 个个体进行交叉。交叉概率太大时，种群中串的更新很快，进而会使高适配值的个体很快被破坏掉；概率太小时，交叉操作很少进行，从而会使搜索停滞不前。

3. 变异概率 M

变异概率是加大种群多样性的重要因素。基于二进制编码的 GA 中，每一代有 $N\times M\times L$ 个基因发生变异（L 为码长），通常一个较低的变异率足以防止整个群体中任一位置的基因一直保持不变。但是，变异概率太小则不会产生新个体，概率太大则使 GA 成为随机搜索。

4. 代沟 G

代沟用于控制每代中种群被替换的比例，即每代有 $N\times(1-G)$ 个父代个体被选中进入下一代种群。$G=100\%$ 意味着所有个体将被替换；$G=50\%$ 意味着一半种群将被替换。标准 GA 中 $G=100\%$，而有些替换策略中 G 值跟新旧个体的适配值好坏有关而且是变化的，如将父代种群的临时种群择优构成下一代种群。

5. 尺度窗口 W

该参数用于作出由目标值到适配值的调整。譬如最大化函数优化问题，个体 x 的适配值可以定义为

$$u(x)=f(x)-f_{\min}$$

其中 $f(x)$ 为其目标值，$f_{\min}$ 为问题的最小目标值。然而，$f_{\min}$ 通常是未知的，许多方法用算法当前进程发现的最小目标值 $f'_{\min}$ 作为替代，但显然定义不同的 $f'_{\min}$ 将影响优良个体的选择压力。Grefenstette(1986)定义了 3 种方案：若 $W=0$ 则首先令 $f'_{\min}$ 为第 1 代种群的最小目标值，在后续的进化中忽视选择目标小于 $f'_{\min}$ 的个体，而当种群中所有个体目标大于 $f'_{\min}$ 时才更新 $f'_{\min}$；若 $0<W<7$，则令 $f'_{\min}$ 为发生在最近 W 代进化中的最小目标值；若 $W=7$，则不做尺度变换操作，即无穷大窗口。

6. 选择策略 S

通常有两种选择策略：其一为纯选择(pure selection)，记 $S=P$，即种群中每一个体根据其适配值作比例选择；其二为保优策略(elitist strategy)，记 $S=E$，即先用纯选择进行选择，然后将 best so far 加入下一代种群，该策略可防止最优解的遗失。

De Jong(1975)对函数优化的标准 GA 的参数做了大量数值研究后认为，最佳 GA 参数为：种群数 50，交叉概率 0.6，变异概率 0.001，代沟 100%，尺度窗口无穷大，采用保优策略。

Grefenstette(1986)则将 GA 的参数选取作为一个优化问题，提出用 GA 优化 GA 参数的二级数值方法。他认为，最佳在线 GA 参数和离线 GA 参数（括号内数值）为：种群数 30(80)，交叉概率 0.95(0.45)，变异概率 0.01(0.01)，代沟 100%(90%)，尺度窗口 1(1)，保优（不保优）。尽管此方法适用范围较广，但工作量较大，

且二级算法本身的参数也有待优化，因此很少得到实际应用。

显然，最佳 GA 参数是依赖于问题的，而上述结论仅针对标准 GA 的函数优化。最近，王凌等(2003)将最佳 GA 参数的确定问题描述为一个随机优化问题，进而利用序优化(ordinal optimization，OO)(Ho，et al，1992)的思想和最优计算量分配(optimal computing budget allocation，OCBA)(Chen，et al，2000)技术来确定最佳参数组合。鉴于该方法论的指导，可以用较少的仿真计算量来以较大的置信度确定适合于所研究问题的最优参数，并且适用于任何类型的优化问题。

总之，尽管 GA 是一种有效的优化算法，但其最优参数的确定本身就是一个极其复杂的优化问题。需要指出的是，由于 GA 是一个动态寻优过程，故一成不变的控制参数未必合理，而自适应控制参数则有助于调整算法的搜索行为和能力。譬如，在进化初期采用较大的遗传概率使得种群有足够的多样性，有助于全局搜索；而在进化后期采用较小的遗传概率来增强局部搜索能力和加速收敛。该问题的解决将有利于 GA 理论和应用的发展，因此研究最优参数的选取和控制仍旧是 GA 研究的重要内容之一，尤其是理论研究。

2.5.4 遗传操作

优胜劣汰是设计 GA 的基本思想，它应在选择、交叉、变异和种群替换等遗传操作中得以体现，并考虑到对算法效率与性能的影响。

1. 种群初始化

鉴于保优 GA 的概率 1 收敛特性，初始种群通常是随机产生的。但考虑到搜索效率和质量，一方面要求尽量使初始种群分散地分布于解空间，另一方面可以采用一些简单方法或规则快速产生一些解作为 GA 的初始个体。如优化 Flow Shop 问题时，可以用 NEH 方法产生一个 GA 初始个体，其余个体则在解空间中均匀选取(王凌等，2003)。

2. 选择操作

选择操作用于避免有效基因的损失，使高性能的个体得以更大的概率生存，从而提高全局收敛性和计算效率。最常用的方法是比例选择、基于排名的选择和锦标赛选择。

(1) 比例选择(也称轮盘赌)。用正比于个体适配值的概率来选择相应的个体，即产生随机数 $\xi\in[0,1]$，若 $\sum_{j=1}^{i-1} f_j \Big/ \sum_{j=1}^{Pop_size} f_j < \xi \leqslant \sum_{j=1}^{i} f_j \Big/ \sum_{j=1}^{Pop_size} f_j$，则选择状态 i 进行复制，其中 f_i 为个体 i 的适配值。

(2) 基于排名的选择。首先将种群中所有个体由好到坏进行排列，然后以一定方式分配给各个体一定的选择概率，具体分配方式不限，如线性方式、非线性方式，但要求越好的个体所分配的概率越大，且所有个体所分配的概率之和为 1。

(3) 锦标赛选择(tournament selection)。首先在父代种群中随机选取 k 个个体，然后令其中适配值或目标值最好的个体为被选中个体。显然，k 的大小影响选择性能。

3. 交叉操作

交叉操作用于组合出新的个体，在解空间中进行有效搜索，同时降低对有效模式的破坏概率。

二进制编码 GA 通常采用单点交叉和多点交叉。

(1) 单点交叉。首先随机确定一个交叉位置，然后对换交叉点后的子串。譬如，父串为(1 0 1 1|0 0 1)和(0 0 1 0|1 1 0)，若单点交叉位置为 4，则后代个体为(1 0 1 1|1 1 0)和(0 0 1 0| 0 0 1)。

(2) 多点交叉。首先随机确定多个交叉位置，然后对换相应的子串。若两点交叉，交叉位置为 2,5，父代个体同上，则后代个体为(1 0| 1 0 1| 0 1)和(0 0| 1 1 0| 1 0)。

十进制编码 GA 的交叉操作类似于二进制编码 GA。

实数编码 GA 通常采用双个体算术交叉或多个体算术交叉。

(1) 双个体算术交叉。针对选中的两个个体进行如下交叉，即 $x_1'=\alpha x_1+(1-\alpha)x_2$，$x_2'=\alpha x_2+(1-\alpha)x_1$，其中随机数 $\alpha\in(0,1)$，x_1，x_2 为父代个体，x_1'，x_2' 为后代个体。

(2) 多个体算术交叉。针对多个选中个体进行如下交叉，即 $x'=\alpha_1x_1+\cdots+\alpha_nx_n$，其中 x_i 为父代个体，x'为后代个体，$\alpha_i\in[0,1]$且 $\sum_{i=1}^{n}\alpha_i=1$。

组合优化中的置换编码 GA 通常采用部分映射交叉、次序交叉、循环交叉、基于位置的交叉、Non-ABEL 群交叉等操作(Goldberg, 1989; Davis, 1991)。

(1) 部分映射交叉(partially mapping crossover, PMX)。首先随机选取两个交叉点，交换父代个体交叉点之间的片段，对于交叉点外的基因，若它不与换过来的片段冲突则保留，若冲突则通过部分映射来确定直到没有冲突的基因，从而获得后代个体。譬如两个父代个体为 $p_1=[2\ 6\ 4|7\ 3\ 5\ 8|9\ 1]$，$p_2=[4\ 5\ 2|1\ 8\ 7\ 6|9\ 3]$，若交叉位置为 3,7，则片段(7 3 5 8)和(1 8 7 6)将交换。对于 p_1 的剩余基因，由于 2 不与(1 8 7 6)冲突则直接填入，6 存在冲突，6 的映射基因为 8 仍存在冲突，8 的映射基因为 3 不存在冲突，则将 3 填入后代个体 q_1 的相应位置。依此类推，得到后代个体 $q_1=[2\ 3\ 4|1\ 8\ 7\ 6|9\ 5]$，类似地可得到 $q_2=[4\ 1\ 2|7\ 3\ 5\ 8|9\ 6]$。PMX 算子一定程度上满足 Holland 图式定理的基本性质，子串能够继承父串中的有效模式。

(2) Non-ABEL 方法。采用如下公式得到后代个体，即 $q_1[i]=p_1[p_2[i]]$，$q_2[i]=p_2[p_1[i]]$。若父代个体同上，则后代个体为 $q_1=[7\ 3\ 6\ 2\ 9\ 8\ 5\ 1\ 4]$，$q_2=[5\ 7\ 1\ 6\ 2\ 8\ 9\ 3\ 4]$。Non-ABEL 群置换操作产生后代方式简单，过分打乱了父串，不利于保留有效模式。

(3) 次序交叉(order crossover, OX)。与 PMX 非常类似，它首先随机确定两个

交叉位置，并交换交叉点之间的片段，并从第2交叉位置起在原先父代个体中删除将从另一父代个体交换过来的基因，然后从第2交叉位置后开始填入剩余基因。譬如，若父代个体和交叉点同上，则片段(7 3 5 8)和(1 8 7 6)将交换，从第2交叉位置起 p_1 删除(1 8 7 6)后剩余(9 2 4 3 5)，然后将其填入 q_1 就得到后代个体为 q_1=[4 3 5|1 8 7 6|9 2]，相应地可得到个体 q_2=[2 1 6|7 3 5 8|9 4]。

(4) 单位置次序交叉(Reeves 表其为 C1，1995)。类似于 OX，但只产生一个交叉位置，然后保留父代个体 p_1 交叉位置前的基因片段，并在另一父代个体 p_2 中删除 p_1 中保留的基因，进而将剩余基因片段填入 p_1 的交叉位置后来达到后代个体 q_1。譬如，若父代个体同前，交叉位置为4，则后代个体为 q_1=[2 6 4 7| 5 1 8 9 3]，q_2=[1 5 2 1| 6 7 3 8 9]。

(5) 线性次序交叉(linear order crossover，LOX)。尽管 Croce 等(1995)用穴(hole)移动的方式来描述 LOX，但其实它就是双位置次序交叉，与 OX 相比仅填入基因起始位置不同。具体而言，首先随机确定两个交叉位置，并交换交叉点之间的片段，并在原先父代个体中删除将从另一父代个体交换过来的基因，然后从第1个基因位置起依次在两交叉位置外填入剩余基因。譬如，若父代个体和交叉点同上，则片段(7 3 5 8)和(1 8 7 6)将交换，在 p_1 中删除(1 8 7 6)后剩余(2 4 3 5 9)，然后将其填入 q_1，就得到后代个体为 q_1=[2 4 3|1 8 7 6| 5 9]，相应地可得到个体 q_2=[4 2 1|7 3 5 8| 6 9]。

(6) 基于位置的交叉(position-based crossover，PX)。PX 与 OX 类似，只是它不再选取连续的基因片段，而是随机选取一些位置，然后交换被选中位置上的基因，并在原先父代个体中删除从另一父代个体交换过来的基因，接着从第1个基因位置起依次在未选中位置填入剩余基因。譬如，若父代个体同前，假设随机选取的位置点为2，3，6，8，则后代为 q_1=[6 5 2 4 3 7 8 9 1]，q_2=[2 6 4 1 8 5 7 9 3]。

(7) 循环交叉(cycle crossover，CX)。将另一个父代个体作为参照，以对当前父代个体中的位置进行重组，先与另一父代个体实现一个循环链，并将对应位置的基因填入相应的位置，循环组成后再将另一个父代个体各位置的基因填入相同的位置。若父代个体同上，循环链为2-3，则后代个体为 q_1=[2 5 4 1 8 7 6 9 3]，q_2=[4 6 2 7 3 5 8 9 1]。

4. 变异操作

当交叉操作产生的后代适配值不再进化且没有达到最优时，就意味着算法的早熟收敛。这种现象的根源在于有效基因的缺损，变异操作一定程度上克服了这种情况，有利于增加种群的多样性。

二进制或十进制编码中通常采用单位置或多位置替换式变异，即用另一种基因替换某一位置或某些位置上原先的基因。

实数编码中通常采用扰动式变异，即对原先个体附加一定机制的扰动来实现变

异，即 $x'=x+\eta\cdot\xi$，其中 x' 和 x 分别为新旧个体，η 为扰动幅度参数，ξ 为随机扰动变量。ξ 可以服从高斯分布、柯西分布、均匀分布，也可以为混沌变量或梯度信息（王凌等，2000）。

组合优化问题中的置换编码 GA 通常采用互换、逆序和插入变异。

(1) 互换操作(SWAP)，即随机交换染色体中两不同基因的位置。

(2) 逆序操作(INV)，即将染色体中两不同随机位置间的基因串逆序。

(3) 插入操作(INS)，即随机选择某个点插入到串中的不同随机位置。

譬如，若状态为(5 4 1 7 9 8 6 2 3)，两随机位置为 2、6，则 SWAP 的结果为(5 8 1 7 9 4 6 2 3)，INV 的结果为(5 8 9 7 1 4 6 2 3)，INS 的结果为(5 8 4 1 7 9 6 2 3)。

5. 替换策略

种群替换策略通常采用整体替换或部分替换。整体替换即将新种群完全覆盖原先种群，而部分替换则用部分新个体替换部分旧个体。譬如，(μ,λ) 策略由 μ 个父代个体产生 λ 个后代个体，然后从后代个体中选取 μ 个最好的个体作为下一代种群；$(\mu+\lambda)$ 策略则从所有 μ 个父代个体和 λ 个后代个体中选取最好的 μ 个个体作为下一代种群。

2.5.5 算法终止条件

GA 的收敛理论说明了 GA 具有概率 1 收敛的极限性质，然而实际算法通常难以实现理论上的收敛条件。因此，追求的目标应该是具有一定优化质量的快速搜索，这显然与算法操作设计和参数选取有关。由于实际应用 GA 时不允许让其无停止地进行搜索，同时问题的最优解也通常未知，因此必须设计一些近似收敛准则来终止算法进程。常用方法包括：给定一个最大进化步数；给定个体评价总数；给定最佳搜索解的最大滞留步数等。

应该说，GA 是一种复杂的非线性随机智能优化计算模型，纯粹用数学方法来预测其运算结果很难。目前为兼顾优化质量和效率所采取的设计方法大多是经验法或试探法，该方面理论还有待更深入地研究与完善。我们认为，计算机仿真和数学分析相结合的研究途径比较可取。

2.6 遗传算法的改进

尽管遗传算法在理论上具有概率 1 的收敛特性，但实际应用时往往出现早熟收敛或收敛缓慢等缺点。因此，设计遗传算法的框架、操作和参数，主要应针对如何设法高效产生或有助于产生优良的个体成员，而这些成员应能够充分表征整个解空间的特性，从而提高算法的搜索效率、避免早熟收敛现象。归纳而言，就是要综合考虑 GA 的探索（exploration）和开发（exploitation）能力、综合考虑种群的全局收敛

(convergence)和分散多样性(diversity)。现今的改进方法,除了采用自适应算法参数外,大都是针对基因操作、种群的宏观操作、基于知识的操作和并行化GA进行的。以下对此作简单综述。

1. 复制操作

在复制操作方面,已有如下改进工作:

- De Jong(1975)研究了回放式随机采样复制,其缺点是选择误差较大;另外他又提出了无回放式随机采样复制以降低选择误差。
- Brindle(1981)提出了一种选择误差更小、操作简单的确定式采样以及无回放式余数随机采样方法。
- Back(1992)提出了与适配值大小和正负无关的均匀排序策略;另外他提出的全局收敛的最优串复制策略,提高了搜索效率,但不适于非线性较强的问题。
- 对于多目标优化,分享(sharing)操作,即对种群内个体的适配值或选择概率进行变换以减小相似个体的选择强度,从而减少相似个体的复制量,以尽可能地保持种群的多样性来达到同时搜索多个区域的目的。

2. 交叉操作

在交叉操作方面,主要有如下改进算子:

- Goldberg(1989)提出了部分映射交叉(PMX)。
- Starkweather 等(1991)提出了增强边缘重组算子(Enhanced Edge Recombination)。
- Davis(1991)提出了序号交叉算子(Order Crossover)和均匀排序交叉算子(Uniform Order-based Crossover)。
- Smith(1985)提出了循环交叉算子(Cycle Crossover)。
- De Jong(1975)提出了单点交叉算子(One-point Crossover)和多点交叉算子。
- Syswerda(1989)提出了双点交叉算子(Two-point Crossover)。

另外,还有置换交叉、算术交叉、启发式交叉等。

3. 变异操作

除前节介绍的常用变异算子外,变异操作主要发展了自适应变异、多级变异等操作。

4. 高级基因操作

除上述常用遗传操作外,还有如下高级基因操作:

- 双倍体和显性遗传(Diploid and Dominance)用以延长曾经适配值高而目前较差的基因块的寿命,并且在变异概率低的情况下也能保持一定的多样性。
- 倒位操作(Inversion)用以助长有用基因块的紧密形式。
- 优先策略(Elitism)用以把目前解群中最好的解直接放入下一代种群中,如此保证各代种群中总会有目前为止最好的解。

- 静态繁殖(Steady-state Reproduction)用以在迭代过程中用部分优质子串来更新部分父串作为下一代种群，以使优质的父串在下一代中得以保留；没有重串的静态繁殖(Steady-state Without Duplication)用以在形成下一代种群时不含重复的串。
- 多倍体结构(Multiple Chromosome Structure)、分离(Segregation)、异位(Translocation)等操作。
- 王凌等(2001, 2002)提出了多交叉和多变异操作的混合优化框架，以及基于模拟退火机制的变异操作。
- 在种群宏观操作方面，主要是引入小生存环境和物种形成(Niche and Spectiation)的思想，这在分类问题中得到了应用。
- 在基于知识的操作方面，主要是将问题的特殊信息与 GA 相结合，包括混合算法的构造以及在遗传算子中增加知识的操作等。

5. 算法结构

在算法结构方面，已开展了如下研究工作：

- Krishnakumar(1989)为克服群体数目大造成计算时间长的缺点，提出了所谓 μGA 的小群体方法，其仿真结果显示了较高的计算效率和适于动态系统优化的潜力，但尚无严格的理论分析。
- Schraudolph 等(1992)针对二进制编码优化精度差的缺点，提出了一种类似于对解空间进行尺度变换的参数动态编码策略，较好地提高了 GA 的精度，但不适于非线性较强的多模型优化问题。
- Androulakis 等(1991)采用实数编码提出了一种扩展遗传搜索算法，把搜索方向作为独立的变量来处理，以解决有约束的优化问题。
- Poths 等(1994)为克服算法早熟收敛的缺点，提出了类似于并行实现思想的基于变迁和人工选择的遗传算法。
- Grefenstette(1981)全面研究了 GA 并行化实现的结构问题，并给出了多种结构形式，主要有同步主-仆方法(Synchronous Master-Slave)、亚同步主-仆方法(Semi-synchronous Master-Slave)、分布式异步并发方法(Distributed Asynchronous Concurrent)、网络方法(Network)。
- Goldberg(1989)提出了基于对象设计 GA 并行结构的思想。
- Muhlenbein 等(1991)用并行遗传算法求解高维多模型函数的全局最小解，从而提供了 GA 求解高度复杂优化问题的有力实例。
- 王凌等(1998, 2001)将 GA 和模拟退火有机地结合起来提出了 GASA 优化框架，有效解决了一系列调度问题和光学仪器的设计问题。相应地，还有 GA 与禁忌搜索、神经网络、混沌、拉氏松弛法、梯度法等的结合。

6. 针对函数优化的改进

对于函数优化,已有如下特殊的改进手段:

- Goldberg(1989)引入分享思想将解空间分成若干子空间,然后在子空间中产生子群体成员分别进行优化,以求得整个问题的解,避免算法只收敛到某个局部解。
- De Jong(1975)提出聚集(Crowding)的思想,根据群成员中的相似性来部分替换群体中的个体成员,从而将一些个体成员分别聚集于各个群集中,然后在各个群集中分别求解问题的局部解,以实现与分享思想相同的目标。然而,这些方法的应用还有一定的限度,对于解是随机分布的情况就不易奏效。
- 孟庆春等(1995)提出门限变换思想,在选择个体成员进入下一代时,引入门限变换函数将某些优良成员周围的成员传到下一代以达到除劣增优的目的,从而避免搜索的盲目性。
- 对带约束优化问题,处理方法主要有:(1)把问题的约束在个体的表示形式中体现出来,并设计专门的遗传算子,使个体所表示的解在 GA 运行过程中始终保持可行性,这种方法最直接,但适用领域有限,算子的设计也较困难;(2)在编码过程中不考虑约束,而在 GA 运行过程中通过检验解的可行性来决定解的弃用,这种方法一般只适用于简单的约束问题;(3)采用惩罚的方法来约束越界问题。

需要指出的是,截至目前为止,采用 GA 求解高维、多约束、多目标的优化问题仍是一个没有较好解决的课题,它的进展将会推动 GA 在许多工程领域的应用。

2.7 免疫遗传算法

本节单独介绍一种改进遗传算法,即基于免疫的改进遗传算法(genetic algorithm based on immunity),它是生命科学中免疫原理与传统遗传算法的结合。算法的核心在于免疫算子的构造,而免疫算子又是通过接种疫苗和免疫选择两个步骤来完成的。同时,在理论上免疫算法是概率1收敛的。

2.7.1 引言

遗传算法是一类基于"产生+测试"方式的迭代搜索算法,尽管该算法在一定条件下具有全局收敛特性,但该算法的交叉、变异、选择等操作一般都是在概率意义下随机进行的,虽保证了种群的群体进化性,但一定程度上不可避免出现退化现象。此外,尽管遗传算法具有通用性的一面,但却忽视了问题特征信息的辅助作用,同时相对固定的遗传操作使得对不同问题的求解缺少灵活性。大量研究表明,仅仅依赖于以遗传算法为代表的进化算法在模拟人类智能化处理事物能力方面还远远不足,还

需要更深入地挖掘和利用人类的智能资源，而免疫遗传算法就是将生命科学中免疫的原理与遗传算法相结合来提高算法的整体性能，并有选择、有目的地利用待求解问题中的一些特征信息来抑制优化过程中退化现象的出现。

2.7.2 免疫遗传算法及其收敛性

1. 免疫遗传算法

类似于生物自然科学的免疫理论，免疫算子分为全免疫(full immunity)和目标免疫(target immunity)，两者分别对应于生命科学中非特异性免疫和特异性免疫。其中，全免疫指种群中每个个体在遗传算子作用后，对其每一环节进行一次免疫操作；目标免疫则指在进行了遗传操作后，经过一定的判断，个体仅在作用点处发生免疫反应。前者主要应用于个体进化的初始阶段，而在进化过程中基本上不发生，否则将很有可能产生"同化"现象；后者一般将伴随进化的全过程，它是免疫的一个基本算子。

实际的操作过程中，首先对所求解的问题(即抗原(antigen))进行具体分析，从中提取最基本的特征信息(即疫苗(vaccine))；其次，对特征信息进行处理，将其转化为求解问题的一种方案(由此方案得到的所有解的集合称为基于上述疫苗所产生的抗体(antibody))；最后，将此方案以适当的形式转化为免疫算子，以实施具体操作。需要说明的是，待求解问题的特征信息往往不止一个，也就是说针对某一特定抗原所能提取的疫苗也可能不止一个，那么在接种疫苗过程中可以随机地选取一种疫苗进行注射，也可将多个疫苗按照一定的逻辑关系进行组合后再注射。总而言之，免疫的思想主要是在合理提取疫苗的基础上，通过接种疫苗和免疫选择两个操作来完成的。前者是为了提高适应度，后者是为了防止种群的退化。

(1) 接种疫苗

对于个体 x，对它接种疫苗是指按照先验知识来修改其某些基因位上的基因，使所得个体以较大的概率具有更高的适应度。

其实现考虑以下两种特殊情况：

其一，若个体 y 的每一基因位上的信息都是错误的，即每一位码都与最佳个体不同，则对任一个体 x，x 转移到 y 的概率为 0；

其二，若个体 x 的每一基因位都是正确的，即 x 已是最佳个体，则 x 以概率 1 转移为 x。

假设种群 $c=(x_1,\cdots,x_{n_0})$，那么，对种群 c 接种疫苗就是指在 c 中按比例 $\alpha(0<\alpha\leqslant 1)$ 随机抽取 $n_\alpha=\alpha n$ 个个体而进行的操作。疫苗是从对问题的先验知识中提炼出来的，它所包含的信息量及其正确性对算法的性能起着重要的作用。

(2) 免疫选择

该操作分两步完成。第一步是免疫检测，即对接种了疫苗的个体进行检测，若其

适应度仍不如父代，则说明在交叉、变异的过程中出现了严重的退化现象。此时，该个体将被父代中所对应的个体所取代。如果子代适应度优于父代，则进行第二步处理。第二步是基于模拟退火算法思想的退火选择操作，即在当前子代种群 $E_k=(x_1,\cdots,x_{n_0})$ 中以概率 $P(x_i)=e^{f(x_i)/T_k}\Big/\sum_{i=1}^{n_0}e^{f(x_i)/T_k}$ 选择个体 x_i 进入新的父代种群，其中 $f(x_i)$ 为 x_i 的适应度，$\{T_k\}$ 为趋于 0 的温度序列。

至此，给出免疫遗传算法如下，其流程如图 2.7.1 所示。

［**免疫遗传算法**］：

步骤 1：随机产生初始父代种群 A_1。

步骤 2：根据先验知识抽取疫苗。

步骤 3：若当前种群中包含了最佳个体，则结束算法；否则进行以下步骤。

步骤 4：对当前第 k 代父代种群 A_k 进行交叉操作，得到种群 B_k。

步骤 5：对种群 B_k 进行变异操作，得到种群 C_k。

步骤 6：对种群 C_k 进行接种疫苗操作，得到种群 D_k。

步骤 7：对种群 D_k 进行免疫选择操作，得到新一代父代种群 A_{k+1}，返回步骤 3。

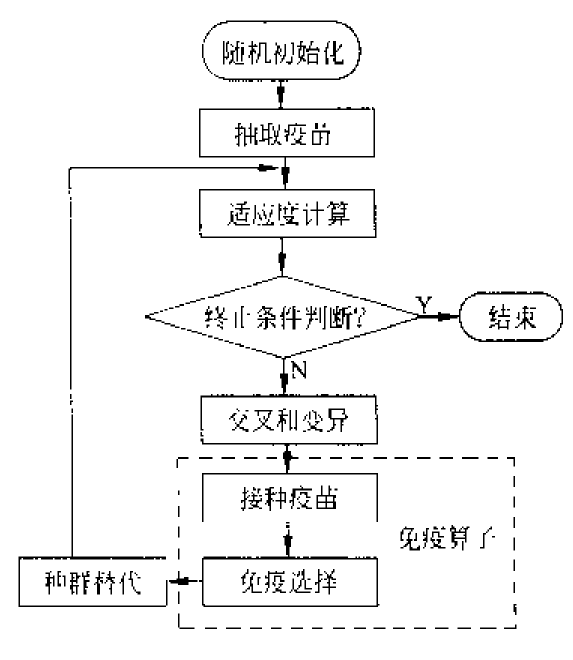


图 2.7.1　免疫遗传算法流程图

2. 免疫遗传算法的收敛性

设种群的规模为 n_0，且在算法中保持不变，种群中所有个体均为 l 位的 q 进制编码，算法中的交叉操作选择一点或多点均可，变异操作是对每个基因位以概率 P_m 相互独立地进行变异。算法的状态转移情况可用随机过程 $A_k\xrightarrow{交叉}B_k\xrightarrow{变异}C_k\xrightarrow{接种疫苗}$

$D_k \xrightarrow{\text{疫苗选择}} A_{k+1}$来表示，其中 A_k 到 D_k 的状态转移构成一个马尔可夫链。设 X 为个体搜索空间，种群可视为状态空间 $S=X^{n_0}$ 中的一个点，$|S|$ 为 S 的规模，$s_i \in S$ 为一个种群，V_k^i 表示随机变量 V 在第 k 代处于状态 s_i，记 $S^* = \{x \in X; f(x) = \max_{x_i \in X} f(x_i)\}$ 为最优状态集。

定义 2.7.1 若对任意初始分布，均有 $\lim_{k\to\infty} \sum_{s_i \cap S^* \neq \emptyset} P(V_k^i) = 1$，则称算法收敛。

该定义表明：所谓算法收敛，即指当算法迭代到足够多的次数以后，群体中包含全局最佳个体的概率接近于 1，也即算法以概率 1 收敛。

定理 2.7.1 免疫遗传算法是概率 1 收敛的，若算法中略去免疫算子，则该算法将不再保证收敛到全局最优值，或者说它是强不收敛的。

证明 参见文献(Jiao and Wang, 2000)。

2.7.3 免疫算子的机理与构造

1. 免疫算子的机理

免疫算子是由接种疫苗和免疫选择两部分操作构成的，其中，疫苗指的是依据人们对待求问题所具备的先验知识而从中提取出的一种基本的特征信息，抗体是指根据这种特征信息而得出的一类解。前者可看作是对待求的最佳个体所能匹配模式(schema)的一种估计，后者则是对这种模式进行匹配而形成的样本。

从对算法的描述中不难发现，疫苗的正确选择对算法的运行效率具有十分重要的意义。它如同遗传算法的编码一样，是免疫操作得以有效发挥作用的基础与保障。但需说明的是，选取疫苗的优劣，生成抗体的好坏，只会严重影响到免疫算子中接种疫苗作用的发挥，不至于涉及到算法的收敛性。因为免疫遗传算法的收敛性，归根结底是由免疫算子中的免疫选择来保证的。

下面考察免疫选择在算法运行过程中所起到的作用。

定理 2.7.2 在免疫选择作用下，若疫苗使抗体适应度得到提高，且高于当前群体的平均适应度，则疫苗所对应的模式将在群体中呈指数级扩散；否则，它将被遏制或呈指数级衰减。

证明 参考遗传算法模式定理的证明，参见文献(Jiao and Wang, 2000)。

可见，免疫选择在加强接种疫苗方面具有积极作用，在消除其负面影响方面具有一定的鲁棒性。考虑到免疫遗传算法的应用对象主要是 NP 类问题，而这类问题在规模较小时一般易于求解，或者说易于发现其局部条件下的求解规律。因此，在选取疫苗时，既可以根据问题的特征信息来制作免疫疫苗，也可在具体分析的基础上考虑降低原问题的规模，增设一些局部条件来简化问题，用简化后的问题求解规律来作为选取疫苗的一种途径。但是，在实际的选取过程中应考虑到：一方面，原问题局域化

处理越彻底，则局部条件下的求解规律就越明显，这时虽然易于获取疫苗，但寻找所有这种疫苗的计算量会显著增加；另一方面，每个疫苗都是利用某一局部信息来探求全局最优解，即估计该解在某一分量上的模式，所以没有必要对每个疫苗做到精确无误。因此一般可以根据对原问题局部化处理的具体情况，选用目前通用的一些迭代优化算法来提取疫苗。

2. 免疫算子的执行算法

为表述方便，令 $a^i_{H,k}$ 为对第 k 代第 i 个个体 a^i_k 接种疫苗后所得到的抗体，P_I 为个体接种疫苗的概率，P_V 为更新疫苗的概率，$V(a^i_k, h_j)$ 表示按模式 h_j 修改个体 a^i_k 上基因的接种疫苗操作，n 和 m 分别为群体和疫苗的规模。那么，在针对某一待求问题而构造和应用免疫算子时所进行的过程如下。

[**免疫算子的构造与应用步骤**]：

步骤 1：抽取疫苗：

(1.1) 分析待求问题，搜集特征信息；

(1.2) 依据特征信息估计特定基因位上的模式 $H=\{h_j; j=1,2,\cdots,m\}$。

步骤 2：令 $k=0, j=0$。

步骤 3：While (Conditions = True)

(3.1) 若 $\{P_V\}$ = True，则 $j=j+1$；

(3.2) $i=0$；

(3.3) for ($i \leqslant n$)

(3.3.1) 接种疫苗：$a^i_{H,k} = V_{(P_I)}(a^i_k, h_j)$；

(3.3.2) 免疫检验：若 $a^i_{H,k} < a^i_{k-1}$，则 $a^i_k = a^i_{k-1}$；否则 $a^i_k = a^i_{H,k}$；

(3.3.3) $i=i+1$；

(3.3.4) 退火选择：$A_{k+1} = S(A_k)$，$k=k+1$。

其中，停机条件可以采用最大迭代次数或统计个体最佳适应度连续不变的最大次数。

3. 免疫疫苗的选取示例

前文简要说明了免疫疫苗的一般选取策略，在此结合旅行商问题（traveling salesman problem，TSP）具体讨论其具体过程与步骤。

(1) 分析待求问题，搜集特征信息

假设在某一时刻，从一城市出发欲前往下一个目标城市，那么，一般首先考虑距离当地路程最近的城市，如果它已被访问过，则可选择剔除该城市之外的距离最小的城市，并以此类推。尽管这种贪婪策略不能保证全局最优，但在仅包含几个城市的一个很小范围内，往往不失为一个较好的策略。当然，能否作为最终的解决方案，还需要进一步的判断。

(2) 根据特征信息制作免疫疫苗

基于上述认识，就 TSP 问题的特点而言，在最终的解决方案中，即在最佳路径里必然而且在很大程度上包括了相邻城市间距离最短的路径。显然，这种特点可作为求解问题时以供参考的一种特征信息或知识，从而视为从问题中抽取疫苗的一种途径。因此，在具体实施过程中，只需使用一般的循环迭代方法找出所有城市的邻近城市即可(当然，某一城市可能是两个或多个城市的邻近城市，也可能都不是)。需要强调的是，疫苗不是一个个体，故不能作为问题的解，它仅仅具备个体在某些基因位上的特征。

(3) 接种疫苗

不失一般性，设 TSP 问题中所有与城市 A_i 距离最近的城市为 A_j，并且二者非直接连接而是处于某一路径的两段：A_{i-1}—A_i—A_{i+1} 和 A_{j-1}—A_j—A_{j+1}，如图 2.7.2 中实线所示。

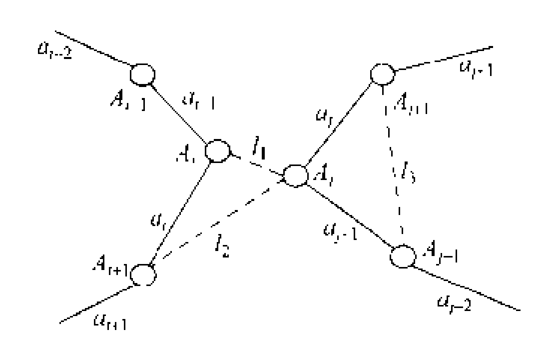


图 2.7.2 TSP 问题的疫苗作用机理示意图

则当前的遍历路径为

$$\pi=\{A_0,\cdots,A_{i-1},A_i,A_{i+1},\cdots,A_{j-1},A_j,A_{j+1},\cdots,A_N\}$$

其对应的路径长度为

$$D_\pi=\sum_{k=1}^{i-1}a_k+a_i+\sum_{k=i+1}^{j-2}a_k+a_{j-1}+a_j+\sum_{k=j+1}^{N}a_k$$

在免疫概率 P_i 发生条件下，对城市 A_i 而言，免疫算子将把其邻近城市 A_j 排列为其下一个城市，而使原先的遍历路径调整为

$$\pi_c=\{A_0,\cdots,A_{i-1},A_i,A_j,A_{i+1},\cdots,A_{j-1},A_{j+1},\cdots,A_N\}$$

相应的路径长度变化为

$$D_{\pi_c}=\sum_{k=1}^{i-1}a_k+l_1+l_2+\sum_{k=i+1}^{j-2}a_k+l_3+\sum_{k=j+1}^{N}a_k$$

比较两路径长度，因为 A_j 是所有城市中(即全局中)与城市 A_i 距离最近的点，在由 A_i—A_j—A_{i+1} 所构成的三角形中，l_1 一定为最短边或次短边(此时 l_2 一定为最短边，因为若 $a_i<l_1$，则与 A_i 最近的城市为 A_{i+1} 而非 A_j)，而在 A_{j-1}，A_j 和 A_{j+1} 之间却不一定具有这个性质。所以，在多数情况下，l_3 较 $a_{j-1}+a_j$ 的减少量要大于 l_1+l_2 较

a_i 的增加量，而且更加重要的是在这一个局部环境内，算子对路径做了一次最佳调整。当然，这次调整究竟能否对整个路径有所贡献，还有待于选择机制的进一步判断。但是，从分析过程中不难得出

$$P(D_{\pi_c} < D_\pi) \gg P(D_{\pi_c} > D_\pi)$$

式中 $P(A)$ 表示事件 A 发生的概率。上述所谓的"调整"过程，即为 TSP 问题求解时基于某一特定疫苗的免疫注射过程。

2.7.4 TSP 的免疫遗传算法

本节以平面 TSP 问题为例，介绍免疫遗传算法的一种实现。

1. 编码与适应度函数

为方便与直观起见，可采用 N 个城市的访问次序为问题的编码。

适应度函数可采用下式计算：

$$f(\pi) = 76.5 \times L\sqrt{N} \cdot D_\pi$$

其中 L 为包含所有城市的最小正方形的边长，D_π 为在排列 π 的路径长度。

2. 交叉与变异算子

采用两点交叉，其中交叉点的位置随机确定。

对于变异操作，算法中加入了对遗传个体基因型特征的继承性和对进一步优化所需个体特征的多样性进行评测的环节，基于此设计一种部分路径变异法。该方法每次选取全长路径的一段，路径子段的起点与终点由评测的结果估算确定。具体操作为采用连续 n 次的调换方式，其中 n 的大小由遗传代数 K 决定，如

$$n = \left[N/M + e^{\alpha K}\right]$$

其中 M 为路径子段的数目，α 为表示快慢程度的常数。

3. 免疫算子

免疫算子包括全免疫和目标免疫两种，对具体问题应视所能提取疫苗的性质而决定采用何种免疫操作。对于 TSP 问题，要找到适用于整个抗原（即全局问题求解）的疫苗极为困难，所以不妨采用目标免疫。具体而言，在求解问题之前先从每个城市点的周围各点中选取一个路径最近的点，以此作为算法执行过程中对该城市点进行目标免疫操作时所注入的疫苗。每次遗传操作后，随机抽取一些个体注射疫苗，然后进行免疫检测，即对接种了疫苗的个体进行检测。若适应度提高，则继续；反之，若其适应度仍不如父代，则说明在交叉、变异的过程中出现了严重的退化现象，这时该个体将被父代中所对应的个体所取代。在选择阶段，先计算其被选中的概率，后进行相应的条件判断。

4. 仿真研究

以著名的 75 城市 TSP 为例，取群体规模为 100，交叉概率在 0.5～0.85 之间，变

异概率在 0.2～0.01 之间，个体接种疫苗概率在 0.2～0.3 之间，更新疫苗概率在 0.5～0.8之间随进化过程自行调整。M 和 α 分别取 10 和 0.04，退温函数为 $T_k=\ln(T_0/k+1)$，$T_0=100$，其中 k 为进化代数。在基本参数保持不变的前提下，对通用遗传算法和免疫算法进行比较，若群体的最佳适应值在连续 100 次迭代中保持不变，则认为结束搜索。同时，计算过程中每隔 10 代记录一次进化结果。图 2.7.3 显示了免疫遗传算法优化 75 城市 TSP 问题的免疫疫苗和优化结果。

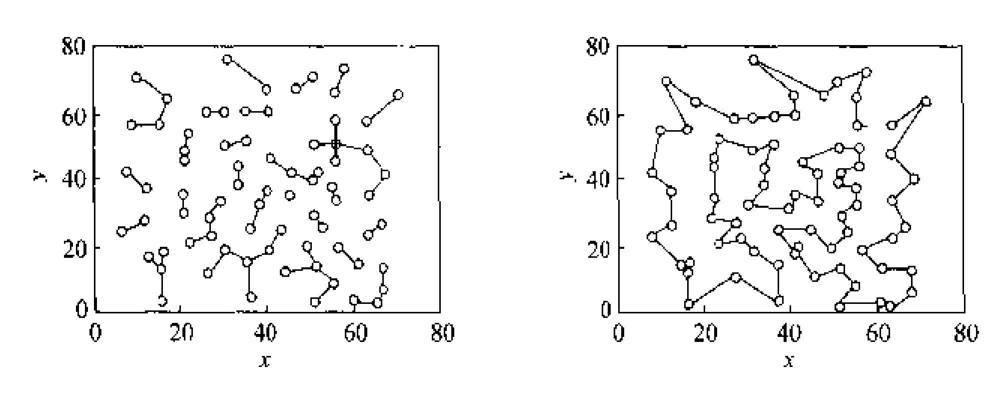


图 2.7.3 优化 75 城市 TSP 问题的免疫疫苗和优化结果

仿真结果表明，免疫遗传算法经 940 代首次出现后来被认定的最佳个体，而通用遗传算法经 3550 代才出现该最佳个体；同时发现，免疫遗传算法对提高搜索效率、消除标准遗传算法在后期的振荡现象具有明显的效果。

2.8 并行遗传算法

GA 的内在并行性在 Holland 提出 GA 时就得到了认识，因此，在并行计算机上实现 GA 是提高算法性能和效率的有效途径。Grefenstette(1981)对 GA 并行化实现的结构问题进行了全面研究，并给出了多种结构形式。本节首先简单介绍最基本的三种并行方案，然后介绍一种可并行实现的基于迁移和人工选择的改进遗传算法(genetic algorithm based on migration and artificial selection, GAMAS)。

2.8.1 同步主仆式

遗传算法的同步主仆式并行执行方案，如图 2.8.1 所示。

在这种并行方式中，一个主过程协调若干个仆过程。其中，主过程控制选择、交叉和变异的执行，仆过程仅执行适配值的计算。这种并行化方式很直观，且易于实现。其存在的两个主要缺点是：若各仆过程计算适配值的时间存在明显差异时，将会造成整个系统长时间的等待；整个系统可靠性较差，对主过程状况的依赖性较大。

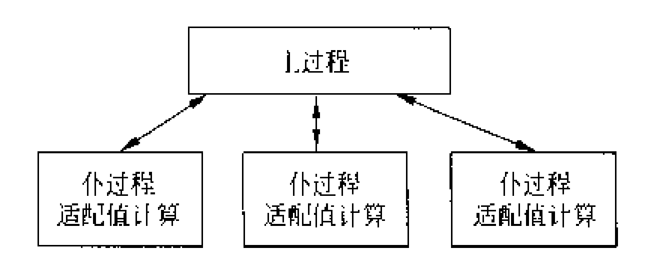


图 2.8.1　同步主仆式并行遗传算法

2.8.2　异步并发式

遗传算法的异步并发式并行执行方案，如图 2.8.2 所示。

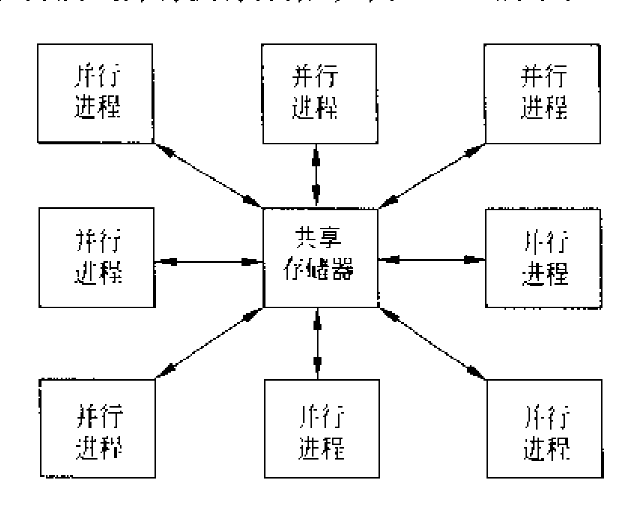


图 2.8.2　异步并发式并行遗传算法

在这种并行方式中，通过存取一个共享存储器，若干个同样的处理机彼此无关地执行各个遗传算子和适配值的计算。只要存在一个并行进程，同时共享存储器可继续运行，则整个系统就可进行有效的处理。显然，这种方式不易实现，但可大大提高系统的可靠性。

2.8.3　网络式

遗传算法的网络式并行执行方案，如图 2.8.3 所示。

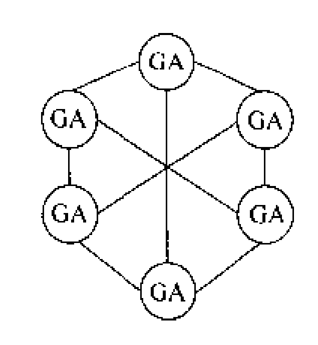


图 2.8.3　网络式并行遗传算法

在这种并行方式中，若干个无关的遗传算法分别在独立的存储器上进行独立的遗传操作和适配值计算，同时各个子群体在每一代中发现的最佳个体通过相应的通信网络传送给其他子群体。与前两种方式相比，由于存在通信时的间断，此方式的连接带宽缩小了。但是，由于各独立过程的自治性，系统的可靠性提高了。

2.8.4 GAMAS 模型

设计 GA 的目标则是快速得到高质量的解，但在 GA 运行过程中其全局收敛与种群多样性存在一定的矛盾，因此均衡整体探索(exploration)和局部开发(exploitation)能力对提高算法性能很关键。若增大选择压力并减小变异概率，则种群将有可能早熟收敛，反之则可能收敛速度缓慢。

Ptees 等在 1994 年提出了一种基于迁移和人工选择的遗传算法 GAMAS，该算法模型很容易并行实现，其结构如图 2.8.4 所示。

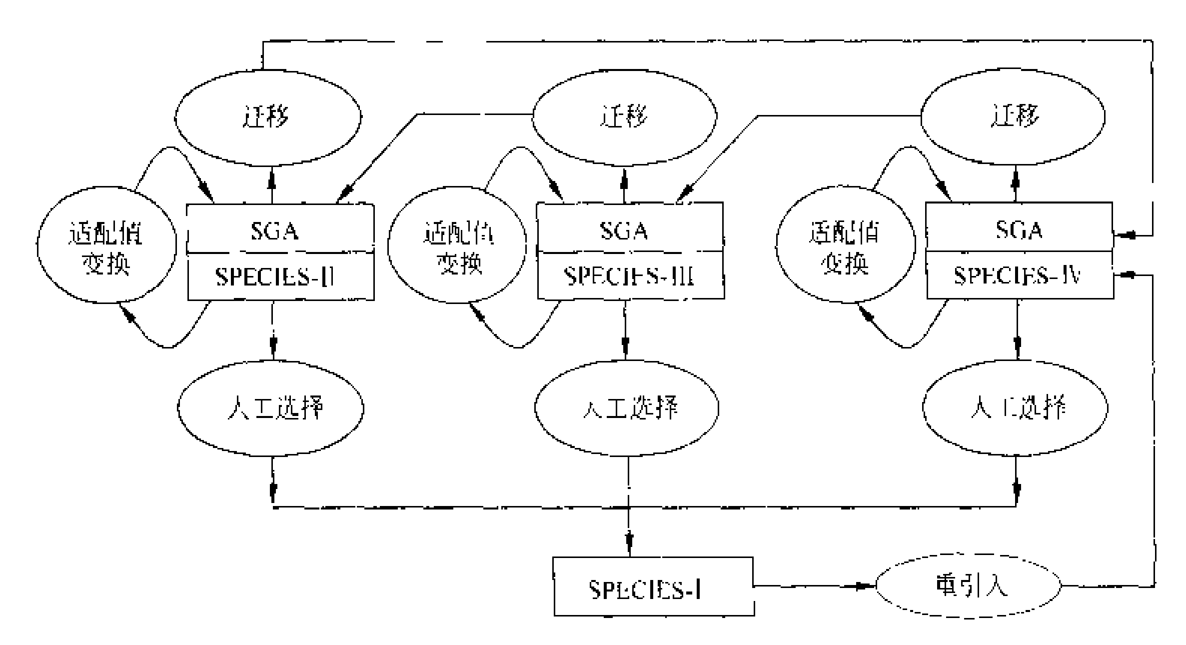


图 2.8.4 GAMAS 模型结构

GAMAS 将种群分解成四个种群数目相同的子种群，分别记为 SPECIES-Ⅰ，SPECIES-Ⅱ，SPECIES-Ⅲ和 SPECIES-Ⅳ，各子种群均有其搜索的倾向性。其中，SPECIES-Ⅰ作为一个子种群本身不进行遗传搜索过程，而是只收集所得到的较好个体再进行人工选择并分配给其他子种群。SPECIES-Ⅱ采用较高的变异概率(譬如 0.05)，以使得搜索侧重于整体探索。SPECIES-Ⅳ采用较低的变异概率(譬如 0.003)，以使得搜索侧重于局部开发。SPECIES-Ⅲ则采用 SEPCIES-Ⅱ和 SPECIES-Ⅳ之间的变异概率(譬如 0.005)。SPECIE-Ⅱ，SPECIE-Ⅲ和 SPECIE-Ⅳ这三个子种群采用 SGA 结构进行微进化(micro-evolution)，利用适配值尺度变换操作来解决适配值差异问题，并采用相同的交叉概率(譬如 0.75)，当然也可不同。如果它们获得了比 SPECIES Ⅰ中个体更好的个体，则用人工选择操作替换 SPECIE Ⅰ中较差的个体，然后再用 SPECIE-Ⅰ中的个体替换 SPECIE-Ⅳ中的所有个体，并且在 SPECIE-Ⅱ，SPECIE-Ⅲ和 SPECIE-Ⅳ之间实行个体迁移，整个过程称为宏进化(macro-evolution)。另外，GAMAS 还采用重初始化策略来增强个体的多样性以防

止早熟收敛。基于模式分类和函数优化的数值仿真验证了GAMAS的有效性，而且很显然，GAMAS是一种并行遗传算法框架。

通过本章对遗传算法理论与实现技术的介绍，我们认识到，GA不只是一种单纯的优化算法，而是一种以生物进化思想为基础的一般方法论，是解决复杂问题的有力工具。GA不是传统的确定性计算工具，动态的复杂问题的求解也不可能是确定的，应建立新的评价标准，这在后文也将给予简单介绍。GA的理论正在深入，应用日趋广泛，但它仅是生物进化系统的简单近似模拟，其本身的发展也是不断进化的过程，理论研究需要引入新的数学工具并吸收生物学的最新成果，应用研究的成败依赖于对GA及其所解决问题的深刻理解。随着GA理论的愈来愈完善，应用领域愈来愈拓宽，在人工生命、其他优化计算技术等领域跟GA相结合后，GA将会发挥其更大的潜力。在后续章节中，我们将分别介绍不同类型调度问题的遗传算法设计。

第3章 Job Shop 调度及其遗传算法

本章首先介绍 Job Shop 调度的描述和若干 Benchmark 问题及其相关的研究结果，然后介绍求解 Job Shop 调度问题的遗传算法的编码设计、遗传操作设计、算法流程设计等，进而介绍 Job Shop 调度的一类有效混合优化策略（遗传退火算法 GASA）和一类模糊 Job Shop 调度问题的遗传算法，最后简单综述近年来有关 Job Shop 调度的遗传算法研究。

3.1 引言

Job Shop 调度问题（简称 JSP）是许多实际生产调度问题的简化模型，是一个典型的 NP-hard 问题，因此其研究具有重要的理论意义和工程价值，它也是目前研究最广泛的一类典型调度问题。JSP 研究 n 个工件在 m 台机器上的加工，已知各操作的加工时间和各工件在各机器上的加工次序约束，要求确定与工艺约束条件相容的各机器上所有工件的加工开始时间或完成时间或加工次序，使加工性能指标达到最优。下面简单给出一个 $n/m/G/C_{\max}$ 调度问题的常用数学描述。

$$\min \max_{1\leqslant k\leqslant m}\{\max_{1\leqslant i\leqslant n} c_{ik}\} \tag{3-1-1}$$

$$\begin{aligned}
\text{s.t.}\quad & c_{ik}-p_{ik}+M(1-a_{ihk})\geqslant c_{ih},\quad i=1,2,\cdots,n,\ h,k=1,2,\cdots,m\\
& c_{jk}-c_{ik}+M(1-x_{ijk})\geqslant p_{jk},\quad i,j=1,2,\cdots,n,\ k=1,2,\cdots,m\\
& c_{ik}\geqslant 0,\quad i=1,2,\cdots,n,\ k=1,2,\cdots,m\\
& x_{ijk}=0\text{ 或 }1,\quad i,j=1,2,\cdots,n,\ k=1,2,\cdots,m
\end{aligned} \tag{3-1-2}$$

其中，式(3-1-1)表示目标函数，即 Makespan；式(3-1-2)表示工艺约束条件决定的各工件的各操作的先后加工顺序以及加工各个工件的各机器的先后顺序。式中，符号 c_{ik} 和 p_{ik} 分别为 i 工件在机器 k 上的完成时间和加工时间；M 是一个足够大的正数；a_{ihk} 和 x_{ijk} 分别为指示系数和指示变量，其意义为

$$a_{ihk}=\begin{cases}1, & \text{若机器 } h \text{ 先于机器 } k \text{ 加工工件 } i\\ 0, & \text{非上述情况}\end{cases} \tag{3-1-3}$$

$$x_{ijk}=\begin{cases}1, & \text{若工件 } i \text{ 先于工件 } j \text{ 在机器 } k \text{ 上加工}\\ 0, & \text{非上述情况}\end{cases} \tag{3-1-4}$$

此外，析取图是描述 JSP 的常用工具。对于 n 个工件、m 台机器（共 N 个操作）的 JSP，所对应的析取图 $G=(V,A,E)$ 如图 3.1.1 所示。其中，V 为所有操作构成的

顶点集，包括 0 和 $N+1$ 两个虚拟操作（分别表示加工开始和结束）；A 为 n 条子边构成的边集，子边（实线）表示某工件按约束条件在所有机器上从开始到结束的加工路径；E 为 m 条子边构成的弧集，子弧（虚线）表示同一机器上加工各操作的连接。

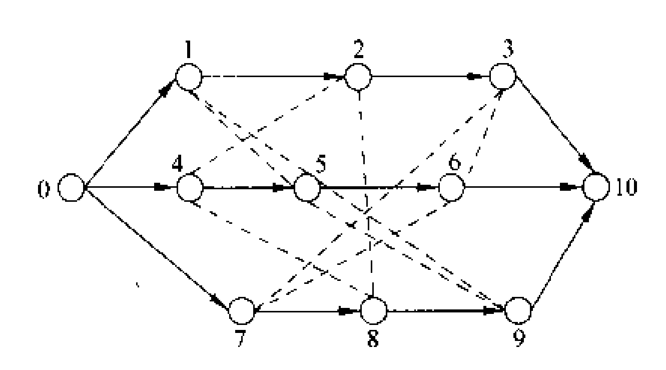


图 3.1.1　3 工件、3 机器 JSP 的析取图

若以最大完成时间为指标，则对 JSP 的求解就归结为找到各弧（即机器）上作为优先决策的各操作的一组顺序（即走向），当同一机器上有多个操作出现冲突时，上述顺序用于决定各操作的先后，最终得到各操作间没有冲突的一个有向非循环图，而其关键路径长度即为最大完成时间。

JSP 的求解远远复杂于旅行商问题和 Flow Shop 调度问题。其原因可归纳如下：

(1) 调度解的编码复杂且多样化，算法的搜索操作多样化；

(2) 解空间容量巨大，n 个工件、m 台机器的问题包含 $(n!)^m$ 种排列；

(3) 存在工艺技术约束条件的限制，必须考虑解的可行性；

(4) 调度指标的计算相对算法的搜索比较费时；

(5) 优化超曲面缺少结构信息，通常复杂且存在多个分布无规则甚至彼此相邻的极小。

对于一个著名的 10 个工件、10 台机器的 JSP（学术界名其为 FT10），从表 3.1.1 对其若干研究结果可见其求解复杂性。对于实际生产调度问题，10 个工件、10 台机器的规模是很小的，由此可预见实际调度问题的求解复杂性。

表 3.1.1　FT10 的若干代表性研究结果

作者和年份	算　法	优化值	时间性能	计算机类型
Lageweg*，1984	列举	930	1h 47min 遇到，持续 9h 6min 无改进	CYBER 170-750
Carlier 等，1989	分支定界	930	产生 22021 个节点，耗时约 5h	Prime 2655
Matsuo 等，1988	Controlled SA	946		
Adams 等，1988	列举型 SB	930	851s	VAX780/11

续表

作者和年份	算　法	优化值	时间性能	计算机类型
Nakano 等，1991	GA	965	种群 1000 进化 150 代	
Van Laarhoven 等，1992	改进型迭代法 SA	1006 930	 57772s	VAX-785
Dell'Amico 等，1993	TS	935		
Croce 等，1995	GA	946	628s	PC 486/25
Dorndoft 等，1995	12 种单一规则 12 种规则混合 基于规则的 GA 移动瓶颈法 SB 部分列举型 SB 基于 SB 的 GA	1191 1088 960 1031 951 938		DEC station 3100

* 参见 Van Laarhoven et al，1992。

对于 JSP 目前的若干求解方法，大量研究表明：

(1) 枚举方法，如分支定界策略，计算量和存储量巨大，难以应用于大规模调度问题。

(2) 基于优先规则的构造性方法和启发式方法，能够快速构造解，但优化质量一般较差。

(3) 移动瓶颈方法能够取得较好的优化质量，但求解过程和算法实施很复杂，且难以移植到其他类型的问题，同时相对其他方法而言比较难理解。

(4) 模拟退火方法、遗传算法、禁忌搜索等方法能够取得较满意的优化质量，但其性能对算法参数有较强依赖性，且优化时间通常较长，而简单地将它们与优先规则相结合的混合方法对优化质量的改善程度比较有限。

(5) 神经网络和蚁群系统等方法优化时间性能较差，而优化质量严重依赖于网络参数，甚至可能产生非法调度。

(6) Lagrangian 松弛法(Hoitomt 等，1993)可以通过松弛和分解的策略降低问题的求解复杂性，但需要合适选取或调整相应的算法参数，同时大多数情况下需要对所得到的解进行再加工，才能够得到可行的较满意的调度。

因此，目前有关 JSP 高效算法的研究与设计仍是 JSP，乃至整个生产调度领域的重要研究内容。迄今，虽然有一些新的方法产生，但其性能还有待进一步研究和验证，而结合已有方法的混合优化算法则一定程度上成为调度问题，乃至复杂优化问题的有效解决途径和目前的国际研究热点，许多国际会议专门列出混合算法专题。本章主要介绍 JSP 的遗传算法设计，至于其他方法的研究读者则可参阅相应文献。

3.2 典型 Job Shop 调度问题

鉴于 JSP 的重要性和代表性，许多研究工作者设计了若干典型问题(benchmarks)，用以测试和比较不同方法的优化性能，下面对其作简单介绍。

1. FT 类

该类问题由 Fisher 和 Thompson 设计(1963)，包括 3 个典型问题，分别称为 FT06，FT10 和 FT20(有些场合也称为 MT06，MT10 和 MT20)，规模分别为 6×6，10×10 和 20×5，前者为工件数，后者为机器数。表 3.2.1 给出了该类问题的若干研究。

表 3.2.1 FT 类问题的若干研究

n	m	问题	最优 Makespan	花费时间/s	计算机	最早取得最优值的研究者
6	6	FT06	55	41	CDC6400	Balas(1969)
10	10	FT10	930	6420	CYBER170-750	Lageweg(1984)
20	5	FT20	1165	151.81	CYBER74	McMahon 等(1975)

2. LA 类

该类问题由 Lawrence(1984)给出，包括 40 个典型问题，命名为 LA1～LA40，对应 8 个不同规模(每一规模包含 5 个问题)，分别为 10×5，15×5，20×5，10×10，15×10，20×10，30×10，15×15，各子类问题也被命名为 F1～F5，G1～G5，A1～A5，B1～B5，C1～C5，D1～D5 和 I1～I5。表 3.2.2 给出了该类问题的若干研究。

表 3.2.2 LA 类问题的若干研究

问　题	最优 Makespan	花费时间/s	计算机	最早取得最优值的研究者
10×5				
LA1(F1)	666	1.26	VAX780/11	Adams 等(1988)
LA2(F2)	655	3.03	VAX780/11	Matsuo 等(1988)
LA3(F3)	597	34.1	VAX780/11	Matsuo 等(1988)
LA4(F4)	590	33.3	VAX780/11	Matsuo 等(1988)
LA5(F5)	593	0.52	VAX780/11	Adams 等(1988)
15×5				
LA6(G1)	926	1.28	VAX780/11	Adams 等(1988)
LA7(G2)	890	1.51	VAX780/11	Adams 等(1988)
LA8(G3)	863	4.52	VAX780/11	Adams 等(1988)
LA9(G4)	951	0.85	VAX780/11	Adams 等(1988)
LA10(G5)	958	14	VAX785	Van Laarhoven 等(1992)

续表

问　题	最优 Makespan	花费时间/s	计 算 机	最早取得最优值的研究者
20×5				
LA11(H1)	1222	2.03	VAX780/11	Adams 等(1988)
LA12(H2)	1039	0.87	VAX780/11	Adams 等(1988)
LA13(H3)	1150	1.23	VAX780/11	Adams 等(1988)
LA14(H4)	1292	0.94	VAX780/11	Adams 等(1988)
LA15(H5)	1207	3.09	VAX780/11	Adams 等(1988)
10×10				
LA16(A1)	945	59	PRIME2655	Carlier 等(1990)
LA17(A2)	784	94	VAX780/11	Matsuo 等(1988)
LA18(A3)	848	106	VAX780/11	Matsuo 等(1988)
LA19(A4)	842	115	VAX780/11	Matsuo 等(1988)
LA20(A5)	902	667	VAX785	Van Laarhoven 等(1992)
15×10				
LA21(B1)	1046	87478	Sparc station ELC	Applegate 等(1991)
LA22(B2)	927	183	VAX780/11	Matsuo 等(1988)
LA23(B3)	1032	225	VAX780/11	Matsuo 等(1988)
LA24(B4)	935	65422	Sparc station ELC	Applegate 等(1991)
LA25(B5)	977	98.2	Sparc station ELC	Applegate 等(1991)
20×10				
LA26(C1)	1218	53	VAX780/11	Matsuo 等(1988)
LA27(C2)	1235	25307	IBM RS6000/320	Carlier 等(1994)
LA28(C3)	1216	305	VAX780/11	Matsuo 等(1988)
LA29(C4)	1152	604800	Pentium 90	Martin(1996)
LA30(C5)	1355	551	VAX780/11	Adams 等(1988)
30×10				
LA31(D1)	1784	38.3	VAX780/11	Adams 等(1988)
LA32(D2)	1850	29.1	VAX780/11	Adams 等(1988)
LA33(D3)	1719	25.6	VAX780/11	Adams 等(1988)
LA34(D4)	1721	27.6	VAX780/11	Adams 等(1988)
LA35(D5)	1888	21.3	VAX780/11	Adams 等(1988)
15×15				
LA36(I1)	1268	1303	PRIME2655	Carlier 等(1990)
LA37(I2)	1397	1577	Sparc station ELC	Applegate 等(1991)
LA38(I3)	1196	165	AT 386 DX	Nowichi 等(1996)
LA39(I4)	1233	6745	Sparc station ELC	Applegate 等(1991)
LA40(I5)	1222	150.1	Sparc station ELC	Applegate 等(1991)

3. ABZ 类

该类问题有 Adams 等(1988)给出,包括 2 个不同规模的 5 个典型问题,其中 ABZ5 和 ABZ6 的规模为 10×10,而 ABZ7、ABZ8 和 ABZ9 的规模为 20×15。表 3.2.3给出了该类问题的若干研究。

表 3.2.3 ABZ 类问题的若干研究

n	m	问题	Makespan 上界(下界)	花费时间/s	计算机	最早取得最优值的研究者
10	10	ABZ5	1234	951.5	SUNSparc 1	Applegate 等(1991)
10	10	ABZ6	943	1101	VAX780/11	Adams 等(1988)
20	15	ABZ7	656	65835	Pentium 90	Martin(1996)
20	15	ABZ8	665(645)	未给出	Pentium 90	Martin(1996)
20	15	ABZ9	679(661)	4949	SUNSparc 330	上界 Balas 等(1998) 下界 Martin(1996)

4. ORB 类

该类问题有 Applegate 等(1991)给出,包括 10 个典型问题,命名为 ORB1~ORB10 其规模为 10×10。表 3.2.4 给出了该类问题的若干研究。

表 3.2.4 ORB 类问题的若干研究

问题	最优 Makespan	花费时间/s	计 算 机	最早取得最优值的研究者
10×10				
ORB1	1059	1482.6	SUNSparc 1	Applegate 等(1991)
ORB2	888	2484.6	SUNSparc 1	Applegate 等(1991)
ORB3	1005	2297.6	SUNSparc 1	Applegate 等(1991)
ORB4	1005	1013.3	SUNSparc 1	Applegate 等(1991)
ORB5	887	526	SUNSparc 1	Applegate 等(1991)
ORB6	1010	未给出	Sparc station ELC	Applegate 等(1991)
ORB7	397	未给出	Sparc station ELC	Applegate 等(1991)
ORB8	899	未给出	Sparc station ELC	Applegate 等(1991)
ORB9	934	未给出	Sparc station ELC	Applegate 等(1991)
ORB10	944	未给出	Sparc station ELC	Applegate 等(1991)

5. SWV 类

该类问题有 Storer 等(1992)给出,包括 4 个不同规模的 20 个典型问题,其中 SWV1-hard~SWV5-hard 的规模为 20×10,SWV6-hard~SWV10-hard 的规模为 20×15、SWV11-hard~SWV15-hard 的规模为 50×10,SWV16-easy~SWV20-easy 的规模也是 50×10。表 3.2.5 给出了该类问题的若干研究。

表 3.2.5 SWV 类问题的若干研究

问 题	Makespan 上界(下界)	花费时间/s	计 算 机	最早取得最优值的研究者
20×10				
SWV1-hard	1407	未给出	Pentium 90	Martin(1996)
SWV2-hard	1475	未给出	Pentium 90	Martin(1996)
SWV3-hard	1398(1369)	未给出	未给出	Vaessens(1996)
SWV4-hard	1483(1450)	2433	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
SWV5-hard	1424	未给出	Pentium 90	Martin(1996)
20×15				
SWV6-hard	1678(1591)	20957	Amd-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV7-hard	1620(1446)	未给出	未给出	Vaessens(1996)
SWV8-hard	1763(1640)	2229	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV9-hard	1663(1604)	未给出	未给出	Vaessens(1996)
SWV10-hard	1767(1631)	23552	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
50×10				
SWV11-hard	2991(2983)	7767	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV12-hard	3003(2972)	22066	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV13-hard	3101	14302	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV14-hard	2968	6112	SUNSparc 330	Balas 等(1998)
SWV15-hard	2904(2885)	30619	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
SWV16-easy	2924	2000 步迭代	SunSystem 4/280	Storer 等(1992)
SWV17-easy	2794	2000 步迭代	SunSystem 4/280	Storer 等(1992)
SWV18-easy	2852	2000 步迭代	SunSystem 4/280	Storer 等(1992)
SWV19-easy	2843	2000 步迭代	SunSystem 4/280	Storer 等(1992)
SWV20-easy	2823	2000 步迭代	SunSystem 4/280	Storer 等(1992)

6. YN 类

该类问题有 Yamada 等(1992)给出,包括 4 个典型问题,命名为 YN1～YN4 其规模为 20×20。表 3.2.6 给出了该类问题的若干研究。

表 3.2.6 YN 类问题的若干研究

问题	Makespan 上界(下界)	花费时间/s	计算机	最早取得最优值的研究者
20×20				
YN1	888(826)	未给出	未给出	上界 Wennink(1995) 下界 Caseau 等(1995)
YN2	909(861)	未给出	未给出	下界 Vaessens(1996) 下界 Caseau 等(1995)
YN3	893(827)	670	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)
YN4	968(918)	23032	AMD-K6/166	上界 Thomsen(1997) 下界 Vaessens(1996)

7. TD 类

该类问题有 Taillard(1993)给出，包括 80 个典型问题，命名为 TD1～TD80，对应8 个不同规模(每一规模包含 10 个问题)，分别为 15×15，20×15，20×20，30×15，30×20，50×15，50×20，100×20。表 3.2.7 给出了该类问题的若干研究。

表 3.2.7 TD 类问题的若干研究

问题	Makespan 上界(下界)	花费时间/s	计 算 机	最早取得最优值的研究者
15×15				
TD1	1231	未给出	未给出	Taillard(1994)
TD2	1244	未给出	AT 386 DX	Nowicki 等(1996)
TD3	1218(1206)	1835	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD4	1175(1170)	未给出	未给出	上界 Wennink(1995) 下界 Vaessens(1996)
TD5	1228(1210)	未给出	未给出	上界 Wennink(1995) 下界 Vaessens(1996)
TD6	1240(1210)	未给出	未给出	上界 Wennink(1995) 下界 Vaessens(1996)
TD7	1228(1223)	未给出	未给出	上界 Taillard(1994) 下界 Vaessens(1996)
TD8	1217(1187)	382	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD9	1274(1247)	646	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD10	1241	1306	SUNSparc 330	Balas 等(1998)

续表

问题	Makespan 上界(下界)	花费时间/s	计算机	最早取得最优值的研究者
20×15				
TD11	1364(1321)	5959	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD12	1367(1321)	5410	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD13	1350(1271)	3169	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD14	1345	未给出	AT 386 DX	Nowicki 等(1996)
TD15	1342(1293)	未给出	未给出	Vaessens(1996)
TD16	1368(1300)	未给出	32 台机器的网络	上界 Aarts(1996) 下界 Vaessens(1996)
TD17	1478(1458)	3107	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD18	1396(1369)	3239	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD19	1341(1276)	5329	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD20	1353(1316)	未给出	未给出	上界 Wennink(1995) 下界 Vaessens(1996)
20×20				
TD21	1647(1539)	未给出	32 台机器的网络	上界 Aarts(1996) 下界 Vaessens(1996)
TD22	1603(1511)	10008	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD23	1558(1472)	8286	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD24	1651(1594)	未给出	32 台机器的网络	上界 Aarts(1996) 下界 Vaessens(1996)
TD25	1598(1496)	未给出	未给出	上界 Taillard(1994) 下界 Vaessens(1996)
TD26	1655(1539)	未给出	未给出	上界 Wennink(1995) 下界 Vaessens(1996)
TD27	1689(1616)	6604	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD28	1615(1591)	6630	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD29	1625(1514)	未给出	32 台机器的网络	上界 Aarts(1996) 下界 Vaessens(1996)

问题	Makespan 上界(下界)	花费时间/s	计 算 机	最早取得最优值的研究者
TD30	1596(1468)	未给出	未给出	Vaessens(1996)
30×15				
TD31	1766(1764)	未给出	AT 386 DX	上界 Nowicki 等(1996) 下界 Taillard(1993)
TD32	1803(1774)	6428	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD33	1796(1778)	11756	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD34	1832(1828)	14277	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD35	2007	未给出	未给出	Taillard(1994)
TD36	1823(1819)	11165	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD37	1784(1771)	12233	SUNSparc 330	上界 Balas 等(1998) 下界 Taillard(1993)
TD38	1681(1673)	6283	SUNSparc 330	上界 Balas 等(1998) 下界 Taillard(1993)
TD39	1798(1795)	3638	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD40	1686(1631)	16628	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
30×20				
TD41	2026(1859)	13198	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD42	1967(1867)	24832	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD43	1881(1809)	12593	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD44	2004(1927)	22045	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD45	2008(1997)	18367	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD46	2040(1940)	23845	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD47	1921(1789)	5617	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD48	1982(1912)	14658	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)

续表

问题	Makespan 上界(下界)	花费时间/s	计 算 机	最早取得最优值的研究者
TD49	1994(1905)	22632	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD50	1951(1807)	未给出	32 台机器的网络	上界 Aarts(1996) 下界 Vaessens(1996)
50×15				
TD51	2760	未给出	未给出	Taillard(1994)
TD52	2756	未给出	未给出	Taillard(1994)
TD53	2717	未给出	未给出	Taillard(1994)
TD54	2839	未给出	未给出	Taillard(1994)
TD55	2679	未给出	AT 386 DX	Nowicki 等(1996)
TD56	2781	未给出	未给出	Taillard(1994)
TD57	2943	未给出	未给出	Taillard(1994)
TD58	2885	未给出	未给出	Taillard(1994)
TD59	2655	未给出	未给出	Taillard(1994)
TD60	2723	未给出	未给出	Taillard(1994)
50×20				
TD61	2868	未给出	AT 386 DX	Nowicki 等(1996)
TD62	2900(2869)	7192	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD63	2755	未给出	AT 386 DX	Nowicki 等(1996)
TD64	2702	未给出	AT 386 DX	Nowicki 等(1996)
TD65	2725	未给出	AT 386 DX	Nowicki 等(1996)
TD66	2845	未给出	AT 386 DX	Nowicki 等(1996)
TD67	2826(2825)	4095	SUNSparc 330	上界 Balas 等(1998) 下界 Vaessens(1996)
TD68	2784	未给出	AT 386 DX	Nowicki 等(1996)
TD69	3071	未给出	AT 386 DX	Nowicki 等(1996)
TD70	2995	未给出	AT 386 DX	Nowicki 等(1996)
100×20				
TD71	5464	未给出	未给出	Taillard(1994)
TD72	5181	未给出	未给出	Taillard(1994)
TD73	5568	未给出	未给出	Taillard(1994)
TD74	5339	未给出	未给出	Taillard(1994)
TD75	5392	未给出	未给出	Taillard(1994)
TD76	5342	未给出	未给出	Taillard(1994)
TD77	5436	未给出	未给出	Taillard(1994)
TD78	5394	未给出	未给出	Taillard(1994)
TD79	5358	未给出	未给出	Taillard(1994)
TD80	5183	未给出	AT 386 DX	Nowicki 等(1996)

8. DMU 类

该类问题有 Demirkol 等(1998)给出,包括 8 个不同规模的 80 个典型问题,命名为 DMU1～DMU80,分别为 20×15,20×20,30×15,30×20,40×15,40×20,50×15,50×20,每一规模包含两组不同加工特性的各 5 个问题。有兴趣的读者可参阅论文(Demirkol,et al,1998)。该文采用 $\alpha/\beta/\gamma$ 方式描述调度问题,同时还给出 320 个以 L_{max} 为指标的典型问题,40 个以 C_{max} 为指标的 Flow Shop 问题和 160 个以 L_{max} 为指标的 Flow Shop 问题,在此不予一一列出。

上述各表中的"花费时间"指算法取得 Makespan 最优值或其上界的 CPU 时间。由于研究人员采用的计算机不同,为了公平比较各方法的计算时间并与所用计算机无关,Jain 等(1999)给出了不同机器的转换因子(见表 3.2.8),从而实际 CPU 时间与转换因子的乘积则可用于公平比较时间性能。读者也可通过比较研究,给出自己所采用计算机的转换因子。

表 3.2.8 若干计算机的转换因子

计算机	转换因子
HP 48GX	0.00081
Apollo DN 3000	0.071
CDC CYBER 170-835	0.47
IBM 4381-23	1.3
DEC Station 5000/200	3.7
IBM RISC Sys/6000-930	15
Cray-2/4-256	48
Fujitsu VP2200/10	127
NEC SX-3/1LR	201
Cray C90	479

上述各类典型 JSP 中,以 FT 类、LA 类和 TA 类调度问题的研究居多。同时,FT 类、LA 类、ABZ 类、ORB 类、SWV 类和 YN 类调度问题可从网址 ftp://mscmga.ms.ic.ac.uk/pub/jobshop1.txt 下载,TA 类调度问题可从网址 ftp://mscmga.ms.ic.ac.uk/pub/jobshop2.txt 下载,而 DMU 类调度问题可从网址 http://gilbreth.ecn.purdue.edu/～uzsoy2/benchmark/problems.html 下载。本书附录列出各调度问题的具体加工数据和工艺约束等内容。另外,上述问题还可分为"简单问题"类和"复杂问题"类,有兴趣的读者可参阅 Jain 等(1999)的综述论文,同时该文还给出了若干典型问题求解方法的优化质量和时间性能。

3.3 Job Shop 调度的遗传算法编码设计

编码问题是设计遗传算法的首要和关键问题。遗传算法的编码技术必须考虑"染色体"的合法性、可行性、有效性以及对问题解空间表征的完全性,求解 JSP 同样如此。鉴于 JSP 的组合特性以及工艺约束性,染色体的 Lamarkian 特性(Whitley 等, 1994)、解码的复杂性、编码的空间特性和存储量的需求是设计遗传算法编码通常要考虑的问题。

(1) 染色体的 Lamarkian 特性。该特性考虑在所设计的编码技术下,染色体的优点(merit)是否可通过一般的遗传操作传到后代种群。如果后代通过遗传操作可有效继承父代的优点,则称所采用的编码具有 Lamarkian 特性,事实上现有许多编码技术具有该特性;如果后代不能够从父代继承任何有用信息,则称所采用的编码不具有 Lamarkian 特性;如果后代所继承的片段中只有部分与父代相同,则称所采用的编码仅具有半 Lamarkian 特性。鉴于遗传算法的优化机制,通常我们希望所设计的编码具有 Lamarkian 特性。

(2) 解码的复杂性。在第 1 章我们曾经介绍了 JSP 的活动调度、半活动调度和非延迟调度的概念。对于正规性能指标,最优调度必然是活动调度。由于非延迟调度集合是活动调度的子集,因此不能够保证非延迟调度就是最优调度;而活动调度又是半活动调度的子集,因此基于半活动调度的搜索空间太大,许多搜索点没有必要。所以,就解码的有效性而言,我们希望 GA 搜索用的码能够解码成活动调度,甚至搜索所采用的码跟活动调度一一对应。就解码复杂性而言,将不需要解码的相应编码归为 0 类复杂性;通过简单映射关系实现解码的相应编码归为 1 类复杂性;通过简单启发式方法才能实现解码的相应编码归为 2 类复杂性;只有通过复杂启发式方法才能实现解码的相应编码归为 3 类复杂性。

(3) 编码的空间特性。编码必须考虑码的可行性、所表征空间的完全性和冗余性。就可行性而言,编码空间通常可分为两类:仅包含可行解的空间,可包含非法解的空间。就完全性而言,有的编码仅表征部分调度空间,而有的编码则可表征整个调度空间,这显然将影响最优调度是否能够得到。就冗余性而言,有的编码使码与调度一一对应,而有的则是 1 到多或多到 1 的关系,这显然影响解码和搜索的效率。

(4) 存储量需求。就 n 个工件 m 台机器的 JSP 而言,通常用码长来反映编码对存储量的需求。称 $n\times m$ 为 GA 染色体的标准长度,即操作的总数量。基于此,可将编码分为三类:其一,码长等于标准长度;其二,码长大于标准长度;其三,码长小于标准长度。

JSP 调度的遗传算法编码可归纳为直接编码和间接编码两种。

- 直接编码将各调度作为状态,通过状态演化达到寻优目的,主要包括基于操作的编码、基于工件的编码、基于工件对关系的编码、基于完成时间的编码、随机键编码等。

- 间接编码将一组工件的分配处理规则作为状态，算法优化的结果是一组最佳的分配规则序列，再由分配规则序列构造调度，主要包括基于优先权规则的编码、基于先后表的编码、基于析取图的编码和基于机器的编码等。

下面以一个 $3/3/G/C_{max}$ JSP 为例对上述各种编码进行介绍。该问题的加工时间和工艺约束见表 3.3.1。以下约定 n 为工件数，m 为机器数。

表 3.3.1　一个 3 工件 3 机器 JSP 的加工数据

项　　目	工件	操作序号		
		1	2	3
操作时间	J_1	3	3	2
	J_2	1	5	3
	J_3	3	2	3
机器顺序	J_1	M_1	M_2	M_3
	J_2	M_1	M_3	M_2
	J_3	M_2	M_1	M_3

3.3.1　基于操作的编码

基于操作的编码（operation-based representation）方式将每个染色体用 $n\times m$ 个代表操作的基因组成，是所有操作的一个排列，其中各工件号均出现 m 次。解码过程是：先将染色体转化为一个有序的操作表，然后基于该表和工艺约束对各操作以最早允许加工时间逐一进行加工，从而产生调度方案。显然，这种解码过程可产生活动调度。

对于上述 $3/3/G/C_{max}$ 例子，假设染色体为[2 1 1 1 2 2 3 3 3]，表 o_{ijk} 为工件 i 的第 j 个操作在第 m 台机器上加工，则对照机器加工顺序的工艺约束，该染色体对应的有序操作表为[o_{211} o_{111} o_{122} o_{133} o_{223} o_{232} o_{312} o_{321} o_{333}]，进而相应的调度如图 3.3.1 所示。

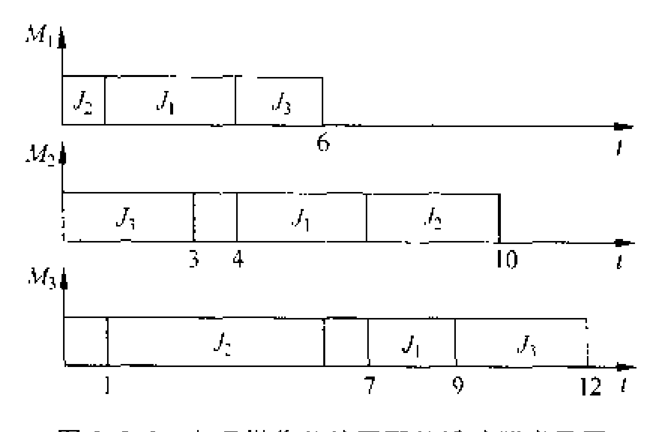
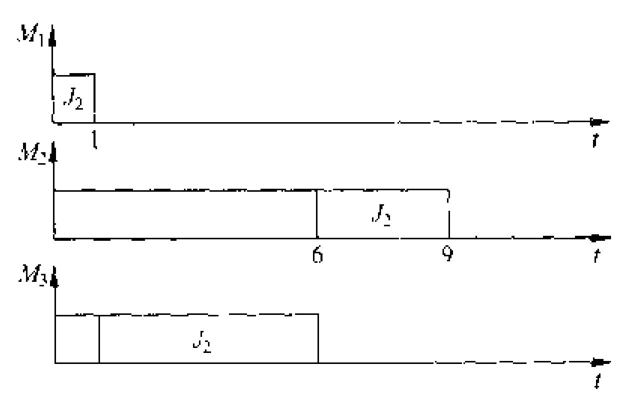


图 3.3.1　基于操作的编码下的活动调度示图

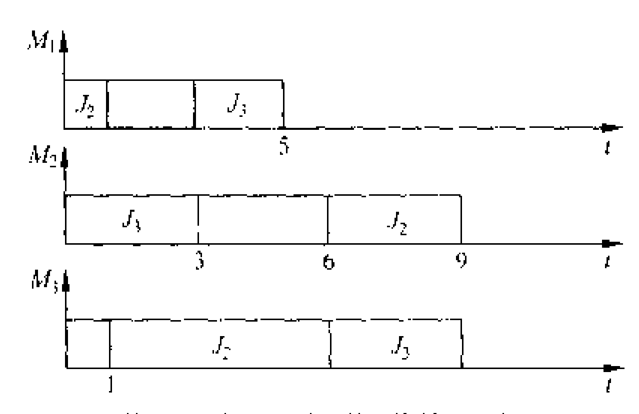
该编码方式的特点可归纳为：半 Lamarkian 性；1 类解码复杂性；任意基因串的置换排列均能表示可行调度（但为保证后代的可行性，遗传操作需特殊设计）；$n \times m$ 标准长度。

3.3.2 基于工件的编码

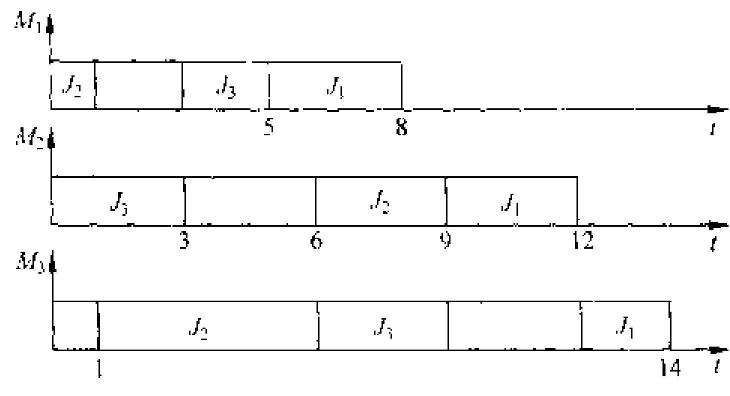
基于工件的编码(job-based representation)方式将每个染色体用 n 个代表工件的基因组成，是所有工件的一个排列。解码过程是：先加工第 1 号工件的所有操作，然后依次以最早允许加工时间加工后面各工件的所有操作。



(a) 基于工件的编码对工件 2 的调度生成示图



(b) 基于工件的编码对工件 3 的调度生成示图



(c) 基于工件的编码对工件 1 的调度生成示图

图 3.3.2 基于工件的编码的调度生成示图

对于上述 3/3/G/C_{max} 例子，假设染色体为[2 3 1]，则先加工工件 2(见图 3.3.2(a))，然后加工工件 3(见图 3.3.2(b))，最后加工工件 1(见图 3.3.2(c))，即为最终调度方案。

该编码方式的特点可归纳为：Lamarkian 性；1 类解码复杂性；任意工件的置换排列均能表示可行调度，但仅能表征部分解空间，不能保证全局最优解的存在性；编码长度小于标准长度。

3.3.3 基于先后表的编码

基于先后表的编码(preference list-based representation)方式将每个染色体用分别对应于 m 台不同机器的 m 个子串构成，各子串是一个长度为 n 的符号串，用于表示一种优先表，各符号表示相应机器上的加工操作。解码过程通过对染色体的仿真得到，即分析机器上当前等待队列的状态并判断是否用先后表来确定调度，也就是说，最先出现在先后表的操作将被选中。

对于上述 3/3/G/C_{max} 例子，假设染色体为[(2 3 1) (1 3 2) (2 1 3)]，其中基因(2 3 1)是机器 1 的先后表，(1 3 2)是机器 2 的先后表，(2 1 3)是机器 3 的先后表。由这些表可以得到，第 1 组优先操作为：工件 2 在机器 1；工件 1 在机器 2；工件 2 在机器 3。考察工艺约束，仅工件 2 在机器 1 可操作，如图 3.3.3(a)。其次可进行的操作为工件 2 在机器 3，如图 3.3.3(b)。至此，当前的优先操作为：工件 3 在机器 1；工件 1 在机器 2 和 3，但它们因不满足工艺顺序均不可进行。因此，察看各表中的下一道优先操作，它们是工件 1 在机器 1、工件 3 在机器 2 和 3。其中，可进行的操作为：工件 1 在机器 1；工件 3 在机器 2，如图 3.3.3(c)。接下来，工件 3 在机器 1 和工件 1 在机器 2，如图 3.3.3(d)。然后是工件 2 在机器 2 和工件 1 在机器 3，如图 3.3.3(e)。最后是工件 3 在机器 3，如图 3.3.3(f)。

该编码方式的特点可归纳为：半 Lamarkian 性；2 类解码复杂性；各状态均能表示可行调度，但难以用遗传操作实行进化；码长为标准长度。

3.3.4 基于工件对关系的编码

Nakano 等(1991)利用二元矩阵表示调度，矩阵由相应机器上工件对的优先关系确定。表 3.3.2 给出了各工件的操作先后顺序和一个可行调度。

表 3.3.2 3 工件 3 机器算例的操作优先顺序和一个可行调度

操作优先顺序				一个可行调度			
工件	机器顺序			机器	工件顺序		
J_1	M_1	M_2	M_3	M_1	J_2	J_1	J_3
J_2	M_1	M_3	M_2	M_2	J_3	J_1	J_2
J_3	M_2	M_1	M_3	M_3	J_2	J_1	J_3

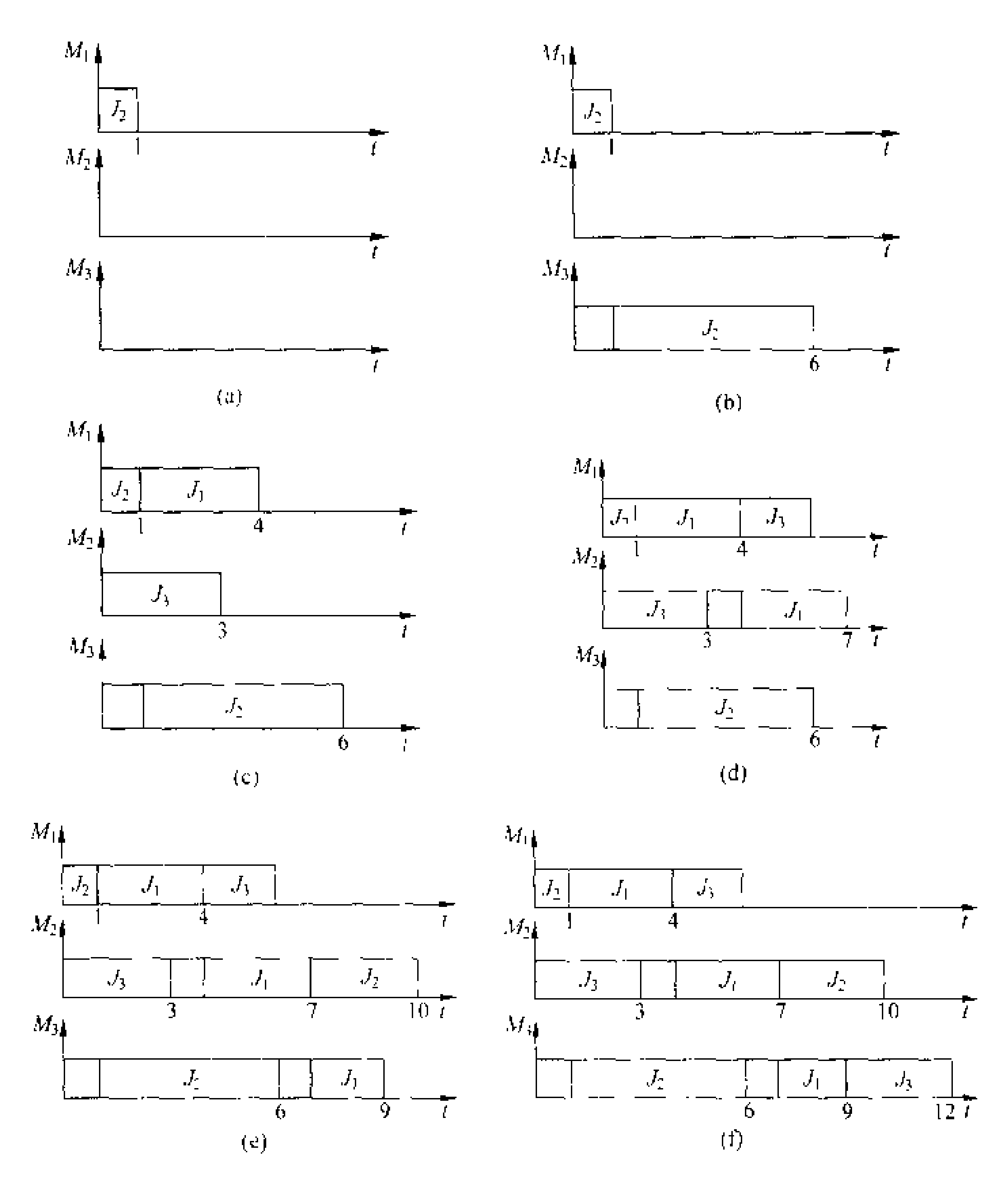


图 3.3.3　基于先后表的编码的调度生成示图

指示工件对优先关系的二值变量 x_{ijk} 定义为：若工件 i 先于工件 j 在机器 m 上加工则取值为 1，否则取值为 0。对于上表给出的调度，(M_1, M_2, M_3) 上工件对 (J_1, J_2) 的先后关系为 $(x_{121}\ x_{122}\ x_{123}) = (0\ 1\ 0)$。工件对 (J_1, J_3) 的先后关系为 $(x_{131}\ x_{132}\ x_{133}) = (1\ 0\ 1)$。对于工件对 (J_2, J_3)，(M_1, M_3, M_2) 的先后关系为 $(x_{231}\ x_{233}\ x_{232}) = (1\ 1\ 0)$。必须注意，工件对的顺序变量 x_{ijk} 必须跟工件 i 的操作的顺序一致。譬如，对于工件对 (J_2, J_3)，J_2 的操作的顺序为(1 3 2)，因此相应变量的排列为 $(x_{231}\ x_{233}\ x_{232})$，而不是 $(x_{231}\ x_{232}\ x_{233})$。由此，表 3.3.2 中可行调度的工件对 (J_1, J_2) 对应机器顺序 (M_1, M_2, M_3)，工件对 (J_2, J_3) 对应机器顺序为 (M_1, M_3, M_2)，工件对

(J_1, J_3)对应机器顺序同于工件对(J_1, J_2)，其二元矩阵码为

$$\begin{bmatrix} x_{121} & x_{122} & x_{123} \\ x_{131} & x_{132} & x_{133} \\ x_{231} & x_{232} & x_{233} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

该编码方式的特点可归纳为：半 Lamarkian 性；1 类解码复杂性；存在较大的冗余；必须考虑合法性；码长大于标准长度。这种编码目前应用较少，也是所有编码方式中最复杂的一种。

3.3.5 基于优先规则的编码

基于优先规则的编码(priority rule-based representation)方式将每个染色体用一个长度为 $n \times m$ 的优先分配规则序列构成，每个基因即为一种优先调度规则。算法的优化结果是一个满意的规则序列，然后依次用这些优先规则产生或修改调度方案。下面首先介绍几种简单常用的优先调度规则。

(1) SPT(shortest processing time)，即选择加工时间最短的操作。

(2) LPT(longest processing time)，即选择加工时间最长的操作。

(3) MWR(most work remaining)，即选择总剩余加工时间最长的工件。

(4) LWR(least work remaining)，即选择总剩余加工时间最短的工件。

(5) MOR(most operations remaining)，即选择总剩余操作数最多的工件。

(6) LOR(least operations remaining)，即选择总剩余操作数最少的工件。

(7) EDD(earliest due date)，即选择交货期最早的工件。

(8) FCFS(first come first served)，即选择同台机器上工件队列中最先的操作。

(9) RANDOM(random)，即随机选择。

记$(p_1, p_2, \cdots, p_{mn})$为一个由优先规则构成的染色体；$PS_t$ 为一个包含 t 个已调度操作的部分调度；S_t 为第 t 步迭代时的可调度操作集合；σ_i 为 S_t 中操作 i 的最早可加工时间；ϕ_i 为 S_t 中操作 i 的最早可完成时间；C_t 为第 t 步迭代时的冲突操作集合。可按以下步骤进行迭代(第 t 步迭代使用规则 p_t)。

[**由基于规则的染色体确定调度方案**]：

步骤 1：令 $t=1$，PS_t 为空，S_t 为没有前道操作的所有操作的集合。

步骤 2：确定 $\phi_t^* = \min_{i \in S_t}\{\phi_i\}$和相应的机器 m^*，若存在多个机器，则任选其一。

步骤 3：在 S_t 中所有欲在 m^* 上加工且 $\sigma_i < \phi_t^*$ 的操作 i 的冲突集合 C_t，用规则 p_t 选择一个操作，若存在多个操作，则任选其一，然后将其尽早加入 PS_t，从而得到 PS_{t+1}。

步骤 4：更新 PS_{t+1}，即从 S_t 中移去选中的操作，将其下道操作加入 S_t，令 $t=t+1$。

步骤 5：返回步骤 2 直到生成一个完全调度。

譬如，仅考虑上述优先规则的前4种，假设染色体为[1 2 2 1 4 4 2 1 3]，其中1表示SPT，2表示LPT，3表示MWR，4表示LWR，则第1步有：$S_1=\{o_{111},o_{211},o_{312}\}$，$\phi_1^*=\min\{3,1,3\}=1$，$m^*=1$，$C_1=\{o_{111},o_{211}\}$。此时，$o_{111}$和$o_{211}$竞争机器1，由于使用SPT规则，故操作$o_{211}$在机器1上先加工（见图3.3.4(a)）。更新数据后有：$S_2=\{o_{111},o_{223},o_{312}\}$，$\phi_2^*=\min\{4,6,3\}=3$，$m^*=2$，$C_2=\{o_{312}\}$，故操作$o_{312}$在机器2上加工（见图3.3.4(b)）。下一步有：$S_3=\{o_{111},o_{223},o_{321}\}$，$\phi_3^*=\min\{4,6,3\}=3$，$m^*=1$，$C_3=\{o_{111},o_{321}\}$。此时，$o_{111}$和$o_{321}$竞争机器1，由于第3个基因为3，即使用LPT规则，故操作o_{111}在机器1上先加工（见图3.3.4(c)）。依次类推，最终得到完全调度（见图3.3.4(d)）。

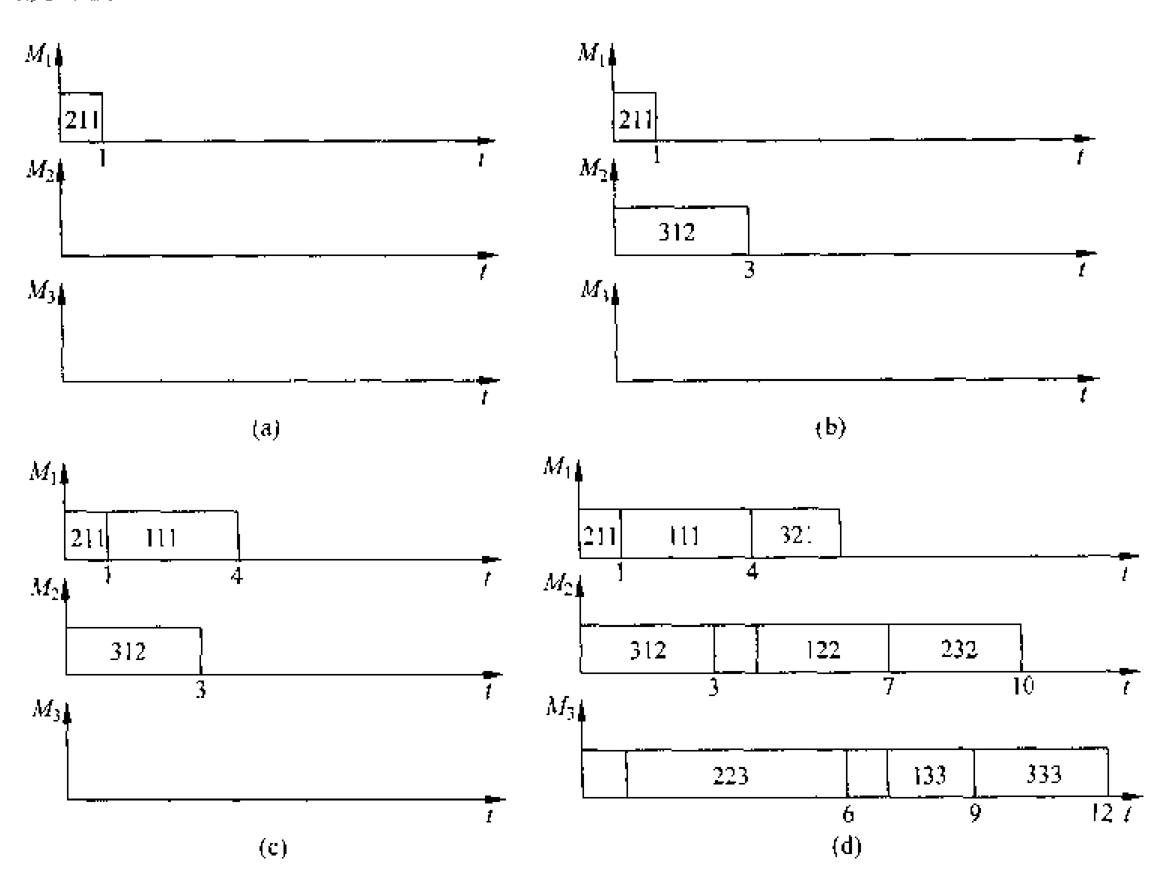


图3.3.4 基于优先规则的编码的调度生成示图

该编码方式的特点可归纳为：Lamarkian性；2类解码复杂性；能保证调度的可行性；码长为标准长度。显然，该编码下GA的优化性能取决于所使用的优先调度规则。

3.3.6 基于析取图的编码

基于析取图的编码(disjunctive graph-based representation)方式将染色体用一个长度为$n\times m$的0 1字符串来表示，该染色体（由各弧的操作顺序组成）作为优先

决策，以决定同台机器上发生操作冲突时各操作的顺序。它也可认为是一种基于工件对关系的编码方式。

3工件3机器JSP问题的析取图见3.1节的图3.1.1，其析取弧集E的各弧的次序表对应一个二值串染色体(见表3.3.3)，其中，e_{ij}表示节点i到节点j的析取弧。其意思为：若设置析取弧的方向为由j到i，则取值为1；反之则取值为0。

表3.3.3　基于析取图的编码方式

析取弧的次序表	e_{15} e_{19} e_{79} e_{24} e_{28} e_{18} e_{36} e_{37} e_{67}
染色体	[0 0 1 1 0 0 0 1 1]

JSP调度就是要得到各机器上各操作的次序，即设置析取弧的方向以得到非循环图来保证没有操作间的先后冲突。显然，任意染色体将导致循环图，即对应不可行解，因此该编码下的染色体不是用来表示调度，而是用作决策的先后，并采用关键路径方法来得到调度。在此过程中，若一台机器上的两个节点(操作)发生冲突，则染色体的相应位用于决定该两操作的先后顺序。

该编码方式的特点可归纳为：半Lamarkian性；3类解码复杂性；能保证调度的可行性；码长为标准长度。

3.3.7　基于完成时间的编码

基于完成时间的编码(completion time-based representation)方式利用各操作完成时间的有序表来表示染色体。

对于上述$3/3/G/C_{max}$例子，染色体可表示为[c_{111}　c_{122}　c_{133}　c_{211}　c_{223}　c_{232}　c_{312}　c_{321}　c_{333}]，其中c_{ijk}表示工件i的第j个操作在机器k上的完成时间。

该编码方式的特点可归纳为：不具有Lamarkian性；0类解码复杂性，即无需解码过程；必须考虑状态的合法性，且需要设计特殊的遗传操作；标准长度。

3.3.8　基于机器的编码

基于机器的编码(machine-based representation)方式将每个染色体用所有机器的排列，并以此通过移动瓶颈方法来构造调度。

移动瓶颈法(shifting bottleneck, SB)是目前解决JSP问题最好的方法之一(Adams等, 1988)，它将各机器逐一排列，每时刻在未排列的机器中定义一台机器为瓶颈，当一台新机器被排列时，所有先前已建立的排列将被再次优化。其中，瓶颈的确定和局部再优化过程基于重复解决某一个单机调度问题(即原问题的一个松弛)而进行。显然，该方法的性能依赖于瓶颈的定义和各瓶颈机器的排列顺序。

令染色体为[m_1, m_2, …, m_m]，M_0为已排列的机器集合，则由该染色体确定一个调度的过程可描述如下，其中步骤2和步骤3详见Adams等的论文(1988)。

[**由基于机器的染色体确定调度方案**]：

步骤 1：令 $M_0=\varnothing$，$i=1$，染色体为$[m_1, m_2, \cdots, m_m]$。

步骤 2：对机器 m_i 进行最优排列，更新 $M_0=M_0\cup\{m_i\}$。

步骤 3：在保持其他机器顺序不变的情况下，对 M_0 中每一个关键机器 m_i 的顺序逐一进行再优化。

步骤 4：令 $i=i+1$。若 $i>m$ 则结束，否则转步骤 2。

该编码方式的特点可归纳为：半 Lamarkian 性；3 类解码复杂性；仅能表征部分解空间，不能保证全局最优解的存在性；码长小于标准长度。

3.3.9 随机键编码

随机键编码(random key representation)将调度解编码成随机数，这些值用作排列键来解码。具体而言，每个基因包括两部分，即{1, 2,…, m}集合中的一个整数以及(0, 1)中的随机分数。任意随机键的整数部分解释为工件的机器分配，对分数部分的排列则提供每一机器上工件的顺序。

对于上述 $3/3/G/C_{max}$ 例子，假设染色体为[1.34 1.09 1.88 2.66 2.91 2.01 3.23 3.21 3.44]，则对机器 1 的随机键的升序排列导致工件顺序为 2—1—3，对机器 2 的随机键的升序排列导致工件顺序为 2—3—1，对机器 3 的随机键的升序排列导致工件顺序为 2—1—3。从而该染色体可解释为一个惟一的有序操作表，即[o_{21} o_{11} o_{31} o_{22} o_{32} o_{12} o_{23} o_{13} o_{33}]，其中 o_{ij} 表示工件 i 在机器 j 上加工。

该编码方式的特点可归纳为：不具有 Lamarkian 性；1 类解码复杂性；表征解空间可能对应非法解；需要对操作的优先顺序作附加处理；标准长度。

3.4 Job Shop 调度的遗传算法操作和框架设计

遗传算法是一种通用的随机优化算法，而 JSP 则是一类特殊的组合优化难题。要使 GA 能够较好地解决 JSP 问题，一方面可对问题进行处理使其适应 GA 的优化，另一方面可对 GA 进行处理使其适应 JSP 的求解，更有效的方法是对 GA 和 JSP 同时作处理使其相互适应，如图 3.4.1 所示。

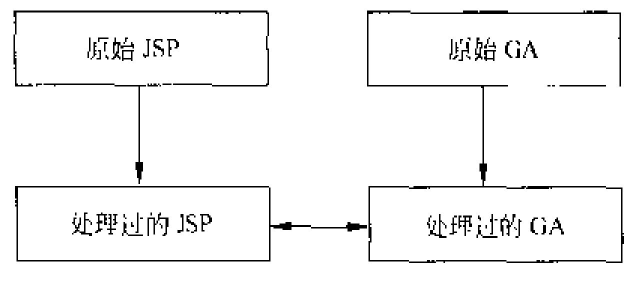


图 3.4.1 GA 优化 JSP 的一种处理方法

这里所谓的处理就是对问题和算法的编码、操作、结构等的处理。由于 JSP 是代表性的组合优化问题之一，置换编码是目前处理这类问题的最常用的方法。因此，目前求解 JSP 的大部分 GA 首先采用遗传操作得到一个合适的置换排列，然后利用一定的方法将其转化为一个调度。需要指出的是，就 GA 的优化特性而言，它是一种通用性算法，没有利用问题的特定信息，并且在局部小空间的"微调"搜索能力有限，因此对于特定问题完全可以将 GA 和其他方法或规则结合使用以取得更好的性能。下面分别介绍求解 JSP 的 GA 的交叉与变异操作以及算法框架的若干设计。鉴于 GA 的选择（或复制）和替换操作适用于任何优化问题，在此不予重复介绍，读者可参阅本书第 2 章相关内容。

3.4.1 JSP 的 GA 交叉与变异操作设计

交叉操作的目的是利用父代个体组合出后代新个体，在尽量降低有效模式破坏概率的基础上对解空间进行高效搜索。进化计算中的遗传算法流派认为，交叉操作是主要的遗传操作，GA 的性能在很大程度上依赖于所使用的交叉操作。鉴于 JSP 的特殊性，GA 必须结合所采用的编码技术设计相应的交叉操作。考虑到置换编码已成为目前 GA 解决调度问题的主流，在此仅介绍若干针对置换编码的交叉操作。读者可阅读有关文献以了解其他类型的交叉操作。

由于 JSP 需要确定各操作的加工顺序，并且要满足各操作进行加工的工艺先后约束，因此其表示方式和求解远比旅行商问题（TSP）复杂。表示 JSP 的置换码通常可分为两类，即单一文字串（pure literal string, PLS）和一般文字串（general literal string, GLS），其中 PLS 由不同的文字排列而成，而 GLS 则允许所排列的文字重复出现。对于前文所介绍的编码技术，基于工件的编码和基于机器的编码采用 PLS；而基于操作的编码、基于优先表的编码和基于先后规则的编码则采用 GLS，其中基于优先表的编码所采用的 GLS 由若干个 PLS 构成。非单一文字串下的交叉操作需要特殊设计，以保证染色体和对应解的可行性与合法性。3.5 节将介绍一种基于操作的编码下的交叉和变异操作的设计，在此仅介绍若干与单一文字串相关的交叉操作的设计。

(1) 部分映射交叉：见本书第 2 章。

(2) 次序交叉：见本书第 2 章。

(3) 基于位置的交叉：见本书第 2 章。

(4) 循环交叉：见本书第 2 章。

(5) 线性次序交叉：见本书第 2 章。

(6) 非 ABEL 群交叉：见本书第 2 章。

(7) 子调度互换交叉：采用 $m \times n$ 维工件矩阵作为编码，每一行代表一台机器上的所有操作的序列或工件的排列（此排列为 PLS），即子调度，该交叉操作就是互换

对应两个父代染色体的矩阵的某同一行上的所有内容。

当然，交叉操作的设计远远不限于这些，只要能够满足后代个体的合法性，任何多样化的设计均可以尝试，甚至可以与启发式算法相结合进行设计。此外，这些操作也可按一定方式结合使用。

在诸多启发式方法中，Giffler-Thompson 算法是一种生成活动调度的枚举方法，其步骤如下。

[**Giffler-Thompson 算法**]：

步骤 1：令 $Q(1)=\{o_{ij}|i=1,\cdots,n;\quad j=1,\cdots,m\}$ 为所有操作的集合；$S(1)$ 为所有工件第 1 道工序的集合。

步骤 2：令 $t=1$。

步骤 3：令 o^* 为满足 $c(o^*)=\min\{c(o_{ij})|o_{ij}\in S(t)\}$ 的操作，m^* 为进行该操作的机器；从集合 $\{o_{im^*}\in S(t);r(o_{im^*})<c(o^*)\}$ 中确定一个操作 o_{im^*}。

步骤 4：生成 $Q(t+1)=Q(t)\backslash\{o_{im^*}\}$。由 $S(t)$ 除去操作 o_{im^*}，并添加工件 i 的下道工序来生成集合 $S(t+1)$。

步骤 5：若 $Q(t+1)$ 为非空，则令 $t=t+1$，并转步骤 3；否则结束算法。

其中，o_{ij} 表示工件 i 在机器 j 上的操作；p_{ij} 表示 o_{ij} 的加工时间；$S(t)$ 表示第 t 步的前一道工序执行时刻的所有未执行操作的集合；$r(o_{ij})$ 表示 $S(t)$ 中的 o_{ij} 对应的工件 i 到达机器 j 的时间；$c(o_{ij})$ 表示 $S(t)$ 中的 o_{ij} 可完成的最早时间，即 $c(o_{ij})=r(o_{ij})+p_{ij}$。

Giffler-Thompson 算法是一种树搜索算法，所生成的活动调度中至少包含调度问题的一个最优解。但应当指出，为保证算法能够生成所有活动调度，在机器选择加工工件时必须考虑尚未到达的部分工件的影响，因此在算法第 3 步中对操作 o^* 的选择必须要在冲突操作集合 $\{o_{im^*}\in S(t);r(o_{im^*})<c(o^*)\}$ 中进行。其中，所谓冲突操作集合指由操作 o^* 相应工件的“理论完成时刻”而非“到达时刻”在机器前排队的所有工件的操作的集合。

下面介绍一种基于 Giffler-Thompson 算法思想的交叉操作。

[**基于 Giffler-Thompson 算法的交叉**]：

步骤 1：令 S 为所有工件第 1 道工序的集合。

步骤 2：确定 $c(o^*)=\min\{c(o_{ij})|o_{ij}\in S\}$，记 m^* 为进行该相应操作的机器，其中 o_{ij} 表示工件 i 的第 j 个操作。

步骤 3：令 G_{m^*} 为 S 需要机器 m^* 进行加工的操作。

步骤 4：从集合 G_{m^*} 中按如下步骤选择一个操作：

(4.1) 产生(0,1)间随机数 ε，若 $\varepsilon<p_m$，则在 G_{m^*} 中任选一个操作记作 o_{ij}^*，其中 p_m 为变异概率。

(4.2) 否则，以相同概率选择两父代个体中的一个个体，记作 p_s，并在

G_{m^*} 的所有操作中找到在个体 p_i 中最早被调度的一个操作，记作 o_{ij}^*。

(4.3) 在后代个体中根据 $c(o_{ij})$ 调度 o_{ij}^*。

步骤 5：更新 S 集合，即 S 除去操作 o_{ij}^*，并添加其下道工序进入集合 S。

步骤 6：若完整的调度方案已生成，则结束；否则转步骤 2。

可见，该方法本质上是基于优先权分配启发式方法的一种算法，与 Giffler Thompson 算法具有相似之处，但也有区别。在算法中，所选取的操作被加入到后代个体的部分调度中，操作的冲突则是基于父代个体的相关信息通过分配优先权来解决。其中，步骤(4.1)通过随机选取一个操作来解决冲突，而步骤(4.2)则是通过给定操作的优先权来解决，即 G_{m^*} 的所有冲突操作中在某个父代个体中最早被调度的那个操作，而这个父代个体是在两父代个体中以等同概率选取的。图 3.4.2 描述了步骤(4.3)对操作的选取。对于后代个体，操作 o_{11}，o_{21}，o_{31} 和 o_{32} 已被调度，可调度的操作集合为 $S=\{o_{12},o_{22},o_{33}\}$。假定冲突操作集合为 $S=\{o_{12},o_{22}\}$，并且父代个体 p_1 被选中，则由于操作 o_{12} 在父代个体 p_1 中是最早被调度的，因此它被调度进入后代个体。

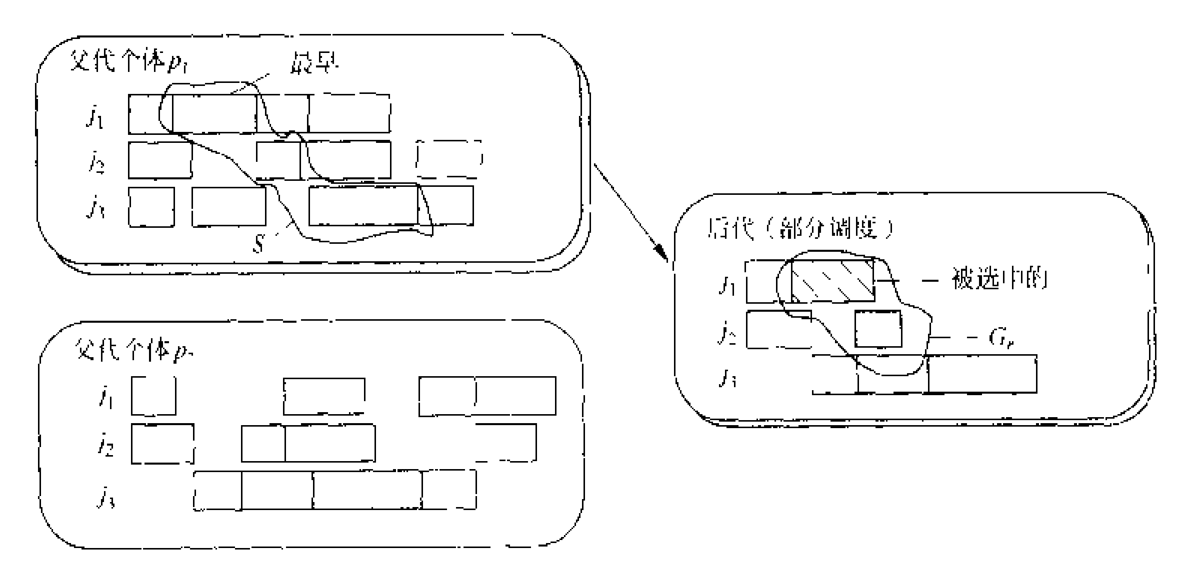


图 3.4.2 基于 Giffler Thompson 算法的交叉示意图

变异操作的目的是通过随机改变染色体的某些基因来引入新个体，增加种群多样性，并在一定程度上进行局部搜索。相对于交叉操作的设计，变异操作的设计更为简单和灵活。常用针对置换编码的变异操作包括：

(1) 互换(swap 或 exchange)：即随机交换若干不同位置上的不同基因。

(2) 逆序(inverse)：即将某两个随机位置间的基因串逆序。

(3) 插入(insert)：即将某一位置上的基因(或某一段位置上的基因串)插入到另一位置之前或之后，若两位置相邻，则该操作也称为漂移(shift)。

当然，变异操作的设计也远远不限于这些，任何能够做到改变个体的邻域操作都

可认为是变异操作，并可加以尝试。另外，也可以将各种不同的变异操作结合使用，以丰富变异的方式，或者说丰富后代的探索方式。

3.4.2 JSP的GA框架设计

诸多研究表明，单独使用遗传算法的性能往往不太满意。原因在于，遗传算法是一类通用的优化算法，对于特定问题其局部优化能力往往比较局限。许多传统的局部搜索方法尽管容易陷入局部极小，但其局部搜索能力很强，这正好能够克服GA的缺点。因此，鉴于GA和传统局部搜索方法或规则的互补特性，许多场合将两者相结合使用，以提高算法的优化质量和效率。

对于调度问题，上述混合思想主要可通过以下3种方式得以体现。

(1) 两阶段方式：利用启发式方法初始化遗传算法，譬如优先规则，以得到具有一定质量的初始种群，如此具有保优特性的GA所得到的解肯定不会差于所采用的传统启发式方法。当然，另一方面也有必要通过一定方式来保持种群的多样性。

(2) 嵌入方式：利用启发式方法作为GA的评价函数，譬如Giffler-Thompson算法和移动瓶颈方法用以将染色体转换为调度，并快速给出其性能指标。需要指出的是，所采用启发式方法的快速性将很大程度上提高GA的整体优化时间性能。

(3) 嵌套方式：利用启发式方法增强或补充GA优化搜索过程，譬如模拟退火算法、禁忌搜索、k-OPT操作和局部趋化性搜索等。许多场合将GA视为在解空间对种群进行全局探索，而将局部搜索方法视为对种群内个体的局部改进。

上述第3种方式是目前研究最多的一类，3.5节介绍的遗传退火方法就属于这一类。这类方式可以用以下一般流程加以描述，其中的局部搜索方法可以多种多样。

[**结合局部搜索的一般GA流程**]：

步骤1：令迭代步数$t=0$。

步骤2：初始化种群$P(t)$。

步骤3：利用局部搜索方法对初始种群进行改进，并评价得到的最终初始种群$P(t)$。

步骤4：若算法终止准则满足则输出最优调度并结束算法，否则继续以下步骤。

步骤5：由遗传操作(选择、交叉和变异)生成临时种群$P'(t)$。

步骤6：利用局部搜索方法对临时种群进行改进，并评价得到的最终临时种群$P'(t)$。

步骤7：利用替换策略得到下一代种群$P(t-1)$。

步骤8：令$t=t+1$，并返回步骤4。

需要强调指出的是，目前算法混合是利用遗传算法等智能优化方法求解复杂优化问题的热点研究方向和途径，但大部分研究成果限于数值分析的结果，在理论方面

还很不完善，尤其是收敛速度分析、Worst-Case 性能分析和有限时间性能分析。

3.5 Job Shop 调度的混合遗传算法

本节介绍求解 $n/m/G/C_{max}$ 调度问题的基于操作的编码的一种混合遗传算法。

3.5.1 编码与解码

为保证编码策略不遗漏问题的全局最优解，使优化操作容易处理状态的可行性和合法性问题，并使算法具有高效的性能，首先来设计一种利用技术约束条件矩阵实施基于操作的编码策略的算法，然后再来设计解码操作将码转化为活动调度，最后设计进行优化的混合策略和优化操作。

1. 编码算法

首先对若干符号进行介绍，然后给出编码算法。

- 机器顺序阵 $\boldsymbol{J}_M$。$\boldsymbol{J}_M(i,j)$ 表示加工 i 工件的第 j 个操作的机器号，$\boldsymbol{J}_M(i,\cdot)$ 表示 i 工件的所有操作按优先顺序加工的各机器号的排列。
- 加工时间阵 $\boldsymbol{T}$。$\boldsymbol{T}(i,j)$ 为 j 工件在 i 机器上的加工时间。
- 工件排列阵 $\boldsymbol{M}_J$。$\boldsymbol{M}_J(i,j)$ 为 i 机器上第 j 次加工的工件号，$\boldsymbol{M}_J(i,\cdot)$ 表示 i 机器上依次加工的各工件的排列。

机器顺序阵即为技术约束矩阵，和加工时间阵一样，也是事先已知的。工件排列阵则是调度的一种表示形式，可用于构造 Gantt 图。

[**编码算法**]：

Repeat

步骤 1：令 $k=1$；

步骤 2：Repeat：

(2.1) 随机产生[1,n]内整数 I 表示某一工件。

(2.2) 由 $\boldsymbol{J}_M$ 得到 I 工件目前尚未加工的优先权最高操作对应的机器 $\boldsymbol{J}_M(I,1)$。

Until $\boldsymbol{J}_M(I,1)\neq 0$。

步骤 3：令 $s[k]=I$，$k=k+1$。

步骤 4：将 $\boldsymbol{J}_M$ 的第 I 行各元依次左移一位，尾部空出的位置填 0。

Until $\boldsymbol{J}_M$ 的所有元均为 0。

上述算法利用技术约束条件（即工件加工的优先次序），是一种基于操作的编码策略，对应状态空间容量为 $(m\times n)!/(n!)^m$。

2. 解码算法

下面，设计将由上述算法产生的码 $s[i]$，$i=1,\cdots,n\times m$ 及其任意置换方式解码

成可行的活动调度策略的算法。

[**活动化解码算法**]：

步骤1：令 $k[i]=1, i=1,\cdots,n$。

步骤2：For $i=1$ to $n\times m$

(2.1) 得到加工工件 $s[i]$ 的机器号 $\boldsymbol{J}_M(s[i],k[s[i]])$。

(2.2) 令 $k[s[i]]=k[s[i]]+1$。

(2.3) 将工件 $s[i]$ 在机器 $\boldsymbol{J}_M(s[i],k[s[i]]-1)$ 上的操作以最早允许加工时间加工，即从零时刻到当前时刻对该机器上的各加工空闲依次判断能否将此工件插入加工。若能，则在空闲中插入加工，并修改该机器上的加工队列；否则，以当前时刻加工该工件，将此工件排在当前队列的末尾。

算法中，工件 I 能在空闲时间段 $[a,b]$ 插入加工的条件为

$$\max(t(I),a)+t_proc\leqslant b$$

其中，a 和 b 分别为空闲起始和终止时刻，$t(I)$ 为工件 I 目前的最早允许加工时间，t_proc 为工件在机器上的加工时间。

若算法中步骤(2.3)采用以当前时刻加工工件的策略，则算法将产生半活动调度，称之为半活动化解码算法。结合编码算法可知，任何可行的半活动调度均能以相同概率产生。因此，问题的最优解必然包含在解空间中。

3. 举例(3工件、2机器)

设机器顺序阵 $\boldsymbol{J}_M=\begin{bmatrix}2,1\\1,2\\1,2\end{bmatrix}$，加工时间阵 $\boldsymbol{T}=\begin{bmatrix}1,3,1\\2,2,1\end{bmatrix}$。

若编码算法产生的随机数序列为[1,3,2,2,1,3]，则相应的半活动调度对应的Gantt图如图3.5.1所示，工件排列阵 $\boldsymbol{M}_s=\begin{bmatrix}3,2,1\\1,2,3\end{bmatrix}$，最大完成时间为7。若作活动化解码处理，所得活动调度对应的Gantt图如图3.5.2所示，工件排列阵为 $\boldsymbol{M}_t=\begin{bmatrix}3,2,1\\1,3,2\end{bmatrix}$，最大完成时间为6(最优解)。

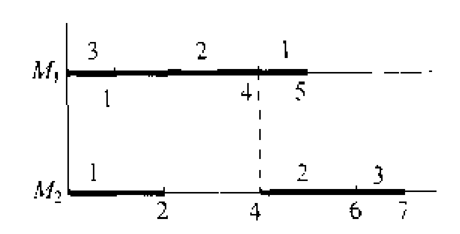


图3.5.1 半活动调度对应的Gantt图

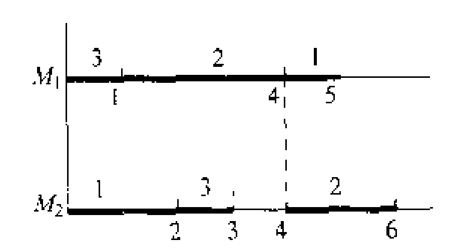


图3.5.2 活动调度对应的Gantt图

3.5.2 混合遗传算法

本节给出 JSP 的结合遗传算法和模拟退火方法的混合遗传算法，以下简称 GASA 混合策略。其中，关于模拟退火算法的简单介绍见本书第 1 章。

1. 算法流程

[JSP 的 GASA 混合优化策略]：

步骤 1：初始化算法参数（种群数目 P_{size}，退温速率 λ，仿真次数等）。

步骤 2：由调度生成算法随机产生初始种群。

步骤 3：经活动化解码计算各个体对应调度的目标值，令最优值 c^* 为当前种群中最优个体的评价值，$k=0$。

步骤 4：判断算法收敛准则是否满足？若是，转步骤 13；否则，转步骤 5。

步骤 5：令 $l=0$。

步骤 6：随机选择个体与种群中的最优个体进行交叉运算，产生两个新个体。若新个体目标值优于 c^*，则更新 c^* 和最优调度；否则，保持 c^* 和最优调度。令 $l=l+1$。

步骤 7：若 $l<P_{size}/2$，则返回步骤 6。

步骤 8：保留原种群和新产生的所有个体中 P_{size} 个最佳个体。

步骤 9：对所有个体进行变异操作，保留 P_{size} 个最佳个体作为 SA 的初始种群，并及时更新 c^* 和最优调度。

步骤 10：对种群中的各个体均进行定步长抽样的模拟退火操作，以概率 $\min[1,\exp(-\Delta/t_k)]$ 接受后代，及时更新 c^* 和最优调度。

步骤 11：令 $k=k+1$，然后进行退温操作 $t_k=\lambda\cdot t_{k-1}$，$\lambda\in(0,1)$，并返回步骤 4。

步骤 12：输出该次仿真所得最优解和寻优时间。

步骤 13：再次进行优化否？若是，则转步骤 2；否则，转步骤 14。

步骤 14：输出多次优化的统计结果，并结束算法。

之所以选择 SA 与 GA 相结合，主要可归纳为以下几方面原因。

(1) 优化机制的融合。理论上，GA 和 SA 两种算法均属于基于概率分布机制的优化算法。不同的是，SA 通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优；GA 则通过概率意义下的基于“优胜劣汰”思想的群体遗传操作来实现优化。对选择优化机制上如此差异的两种算法进行混合，有利于丰富优化过程中的搜索行为，增强全局和局部意义下的搜索能力和效率。

(2) 优化结构的互补。SA 算法采用串行优化结构，而 GA 采用群体并行搜索。两者相结合，能够使 SA 成为并行 SA 算法，提高其优化性能；同时 SA 作为一种自适

应变概率的变异操作,增强和补充了GA的进化能力。

(3) 优化操作的结合。SA算法的状态产生和接受操作每一时刻仅保留一个解,缺乏冗余和历史搜索信息;而GA的复制操作则能够在下一代中保留种群中的优良个体,交叉操作能够使后代在一定程度上继承父代的优良模式,变异操作能够加强种群中个体的多样性。这些不同作用的优化操作相结合,丰富了优化过程中的邻域搜索结构,增强了全空间的搜索能力。

(4) 优化行为的互补。由于复制操作对当前种群外的解空间无探索能力,种群中各个体分布"畸形"时交叉操作的进化能力有限,小概率变异操作很难增加种群的多样性。所以,一方面,若算法收敛准则设计不好,GA经常会出现进化缓慢或"早熟"收敛的现象;另一方面,SA的优化行为对退温历程具有很强的依赖性,而理论上的全局收敛对退温历程的限制条件很苛刻,因而SA优化时间性能较差。若两种算法结合,那么,SA的两准则可控制算法收敛性以避免出现"早熟"收敛现象,并行化的抽样过程可提高算法的优化时间性能。

(5) 削弱参数选择的苛刻性。SA和GA对算法参数具有很强的依赖性,参数选择不合适将严重影响优化性能。SA的收敛条件导致算法参数选择较为苛刻,甚至不实用;而GA的参数又没有明确的选择指导,设计算法时均要通过大量的试验和经验来确定。GA和SA的相混合,使算法各方面的搜索能力均有提高,因此对算法参数的选择不必过分严格。

总之,上述混合策略具有以下特点:

(1) GASA混合策略是标准GA,SA以及并行SA算法的一个统一结构。若在GASA混合策略中移去有关SA的操作,则混合策略转化为GA算法;若移去GA的进化操作,则转化为并行SA算法;进一步,若置种群数为1,则转化为标准SA算法。

(2) GASA混合策略本质上是一个两层并行搜索结构。进程层次上,混合算法在各温度下串行地依次进行GA和SA搜索,是一种两层串行结构。其中,SA的初始解来自GA的进化结果,SA经Metropolis抽样过程得到的解又成为GA进一步进化的初始种群。空间层次上,GA提供了并行搜索结构,使SA转化成为并行SA算法,因此混合算法始终进行群体并行优化。

(3) GASA混合策略利用了不同的邻域搜索结构。混合算法结合了GA和SA搜索,优化过程中包含了GA的复制、交叉、变异和SA的状态产生函数等不同的邻域搜索结构。复制操作有利于优化过程中产生优良模态的冗余信息,交叉操作有利于后代继承父代的优良模式,高温下的SA操作有利于优化过程中状态的全局大范围迁移,变异和低温下的SA操作有利于优化过程中状态的局部小范围趋化性移动,所有这些都有助于增强算法在解空间中的探索能力和效率。

(4) GASA混合策略的搜索行为是可控的。混合策略的搜索行为,可通过退温历程(即初温、退温函数、抽样次数)加以控制。控制初温,可控制算法的初始搜索行

为;控制温度的高低,可控制算法突跳能力的强弱,高温下的强突跳性有利于避免陷入局部极小,低温下的趋化性寻优有利于提高局部搜索能力;控制温度的下降速率,可控制突跳能力的下降幅度,影响搜索过程的平滑性;控制抽样次数,可控制各温度下搜索能力,影响搜索过程对应的齐次马尔可夫链的平稳概率分布。这种可控性增强了克服 GA 易"早熟"收敛的能力。算法实施时,退温历程还可引入可变抽样次数、"重升温"等高级技术。

(5) GASA 混合策略利用了双重准则。理论上,抽样稳定和算法终止准则均由收敛条件决定。但是,这些条件往往不实用。在设计算法时,抽样稳定准则可用以判定各温度下算法的搜索行为和性能,也是混合算法中由 SA 切换到 GA 的条件;算法终止准则可用以判定算法优化性能的变化趋势和最终优化性能。两者结合可同时控制算法的优化性能和效率。

由此可见,在优化机制、结构和行为上,GASA 混合优化策略均结合了 GA 和 SA 的特点,使两种算法的搜索能力得到相互补充,弥补了各自的弱点。因此,它是一种优化能力、效率和可靠性较高的优化方法。

2. 算法操作

- 初温。初温通过式 $t_0=-(c_{worst}-c_{best})/\ln(p_r)$ 确定,其中 c_{best},c_{worst} 分别为初始种群中最佳和最差个体的目标值,p_r 为设置的两者相对接受概率。
- 交叉操作。基于操作的 JSP 编码策略下,个体中存在许多相同"基因",因此难以应用传统处理置换排列的交叉算子,如 PMX,LOX 等。为了使交叉操作具有快速进化的能力,并保证子代个体的可行性,在此设计一种简单的基于互补集的顺序选择法。

[**交叉操作**]:

步骤 1:把 $1\sim n$ 自然数集随机分成两个互补集合 S_1 和 S_2。

步骤 2:从前到后顺序选择父代个体 A_1 中属于 S_1 的基因,以及父代个体 A_2 中属于 S_2 的基因,构成子代个体 B_1。

步骤 3:类似步骤 2,从前到后顺序选择父代个体 A_1 中属于 S_2 的基因,以及父代个体 A_2 中属于 S_1 的基因,构成子代个体 B_2。

- 变异操作。逆序 INV 操作。
- SA 状态产生函数。互换 SWAP 操作。
- SA 退温函数。指数退温,即 $t_{k+1}=\lambda\cdot t_k$。
- 不采用适配值信息,而是直接基于个体的目标值进行遗传操作。

3.5.3 仿真结果与比较

为了考察 GASA 混合策略的性能,以 Borland C++ 为仿真环境,采用 64M RAM

的 Pentium Ⅱ/350 计算机，选取如下算法参数：种群数 P_{size} 取 20，初温下的最差接受概率 p_r 取 0.1，退温速率取 0.9，抽样步数取 $n\times m$，以最优值连续 30 次退温均保持不变为算法终止条件。对 11 个不同规模的典型算例(这些算例的加工数据和工艺约束见附录 1)作如下数值仿真研究：

(1) 对各算例均作 20 次随机仿真，对 GASA 的各项优化指标与 SA 和 GA 作比较，统计结果如表 3.5.1 所示。

表 3.5.1 GASA、GA 和 SA 的性能比较

Problem	n,m	c^{**}	GASA			SA			GA		
			c^*	%	t	c^*	%	$\bar{t}$	c^*	%	t
FT06	6, 6	55	**55**	**0**	**2.12**	55	0	0.16	55	3.273	0.12
LA01	10, 5	666	**666**	**0**	**5.72**	666	0	0.48	666	3.048	0.3
LA06	15, 5	926	**926**	**0**	**12.29**	926	0	0.63	926	0	0.37
LA11	20, 5	1222	**1222**	**0**	**25.94**	1222	0	1.47	1222	0	0.6
FT10	10, 10	930	**930**	**2.548**	**44.37**	939	6.645	2.42	997	11.871	0.5
FT20	20, 5	1165	**1165**	**1.221**	**90.17**	1227	8.326	4.14	1247	12.403	1.19
LA16	10, 10	945	**945**	**1.005**	**29.43**	979	4.169	1.84	979	6.032	0.61
LA21	15, 10	1046	**1058**	**2.590**	**184.46**	1083	7.486	8.92	1156	14.56	1.07
LA26	20, 10	1218	**1218**	**0.441**	**417.88**	1253	5.099	19.85	1328	12.783	2.48
LA31	30, 10	1784	**1784**	**0**	**546.21**	1784	0.392	36.92	1836	4.826	3.62
LA36	15, 15	1268	**1292**	**3.059**	**348.11**	1321	5.801	16.7	1384	11.908	2.4

(2) 对各算法均能得到最优解的算例，对最优解的首达时间作比较，其结果见表 3.5.2 所示。

表 3.5.2 对简单问题的最优解首达时间(秒)比较

Problem	n,m	t_e(GASA)	t_e(SA)	t_e(GA)
FT06	6, 6	**0.034**	0.133	0.108
LA01	10, 5	**0.049**	0.353	0.247
LA06	15, 5	**0.015**	0.049	0.092
LA11	20, 5	**0.034**	0.122	0.233

(3) 以表 3.5.1 中 GASA 相同的时间测试 SA 和 GA 求解复杂问题的性能，其结果见表 3.5.3。

表 3.5.3 SA 和 GA 以 GASA 相同时间(表 3.5.1)求解复杂问题的性能

Problem	n, m	SA		GA	
		c^*	%	c^*	%
FT10	10, 10	937	3.355	980	10.215
FT20	20, 5	1178	3.271	1247	11.442
LA16	10, 10	946	1.101	956	4.44
LA21	15, 10	1079	4.73	1142	11.577
LA26	20, 10	1218	0.562	1299	10.197
LA31	30, 10	1784	0	1785	1.822
LA36	15, 15	1293	3.121	1359	9.621

(4) 将 GASA 算法的最优性能跟 JSP 算法研究的若干权威文献的结果作比较，其结果见表 3.5.4。

表 3.5.4 GASA 混合算法与文献结果的比较

Problem	n, m	c^{**}	GASA Best	GA Best	TS Best	SA Best	SB Best
FT06	6, 6	55	**55**	55	55	55	55
LA01	10, 5	666	**666**	666	666	666	666
LA06	15, 5	926	**926**	926	926	926	926
LA11	20, 5	1222	**1222**	1222	1222	1222	1222
FT10	10, 10	930	**930**	946	935	930	930
FT20	20, 5	1165	**1165**	1178	1165	1165	1178
LA16	10, 10	945	**945**	979	945	956	978
LA21	15, 10	1046	**1058**	1097	1048	1063	1084
LA25	20, 10	1218	**1218**	1231	1218	1218	1224
LA31	30, 10	1784	**1784**	1784	1784	1784	1784
LA36	15, 15	1268	**1292**	1305	1278	1293	1305

注：上述各表中，n, m 分别为工件数和机器数；c^{**} 为问题的最优值；c^* 为算法 20 次仿真得到的最优值；$\bar{t}$ 为20 次随机仿真的平均 CPU 时间；% 为平均优化值相对 c^{**} 的偏差；t_e 为最优解的首达时间。

通过对上述典型算例的仿真研究，可对 JSP 的混合优化策略得到如下的一些结论。

(1) 对于复杂问题，GASA 的优化性能较 SA 和 GA 有很大幅度的提高，且对大

部分算例均能得到最优值。

(2) 对于复杂问题,GASA 的鲁棒性能较 SA 和 GA 有很大幅度的提高。

(3) 对于简单问题,各算法均能得到最优值,但 GASA 搜索到最优解的首达时间最短,即混合策略发现最优解的效率最高。

(4) 与文献结果相比较,混合算法对各算例的最优性能均优于 SA,GA 和 SB 算法;与 TS 相比,GASA 仅对 LA21 和 LA36 的最优性能略次,但对 MT10 的性能要好。

需要说明的是,各文献算法对参数具有很强的依赖性,且波动性能较差,而混合算法在这方面体现出了较大的优越性。同时,就时间性能而言,尽管 GASA 需要花费较多的优化时间,但这一方面是算法终止条件判断的结果,另一方面也是为避免早熟收敛而付出的代价。然而,单一算法即使以混合策略相同时间优化,也达不到混合策略的优化效果。或者说,单一算法要实现混合策略的优化效果,需要付出更多的搜索时间,甚至根本达不到混合策略的优化能力。由于单一算法的性能需要合适的参数作保证,因此算法的混合是放松对参数苛刻选择条件的有效途径。

总之,对于复杂的调度问题,要取得较满意的性能,单纯的遗传算法需要在结构和操作上作改进,尤其是吸收或结合其他方法的优点,进行多阶段串行或同步并行搜索。而 GASA 对 JSP 问题的有效求解,则说明了混合优化策略是解决复杂调度问题的有效而可靠的优选方法,这也是目前国际上的热点研究方向。

附:下面给出 FT10 的加工时间阵、技术约束阵,以及 GASA 算法所得的最优工件排序阵及其对应的最优加工 Gantt 图(见图 3.5.3)。同时,该调度问题的 Gantt 图也明显地反映了此问题的求解复杂性。

$$
\boldsymbol{T}^{\mathrm{T}}=\begin{bmatrix}
29 & 78 & 9 & 36 & 49 & 11 & 62 & 56 & 44 & 21\\
43 & 28 & 90 & 69 & 75 & 46 & 46 & 72 & 30 & 11\\
85 & 91 & 74 & 39 & 33 & 10 & 89 & 12 & 90 & 45\\
71 & 81 & 95 & 98 & 99 & 43 & 9 & 85 & 52 & 22\\
6 & 22 & 14 & 26 & 69 & 61 & 53 & 49 & 21 & 72\\
47 & 2 & 84 & 95 & 6 & 52 & 65 & 25 & 48 & 72\\
37 & 46 & 13 & 61 & 55 & 21 & 32 & 30 & 89 & 32\\
86 & 46 & 31 & 79 & 32 & 74 & 88 & 36 & 19 & 48\\
76 & 69 & 85 & 76 & 26 & 51 & 40 & 89 & 74 & 11\\
13 & 85 & 61 & 52 & 90 & 47 & 7 & 45 & 64 & 76
\end{bmatrix}
$$

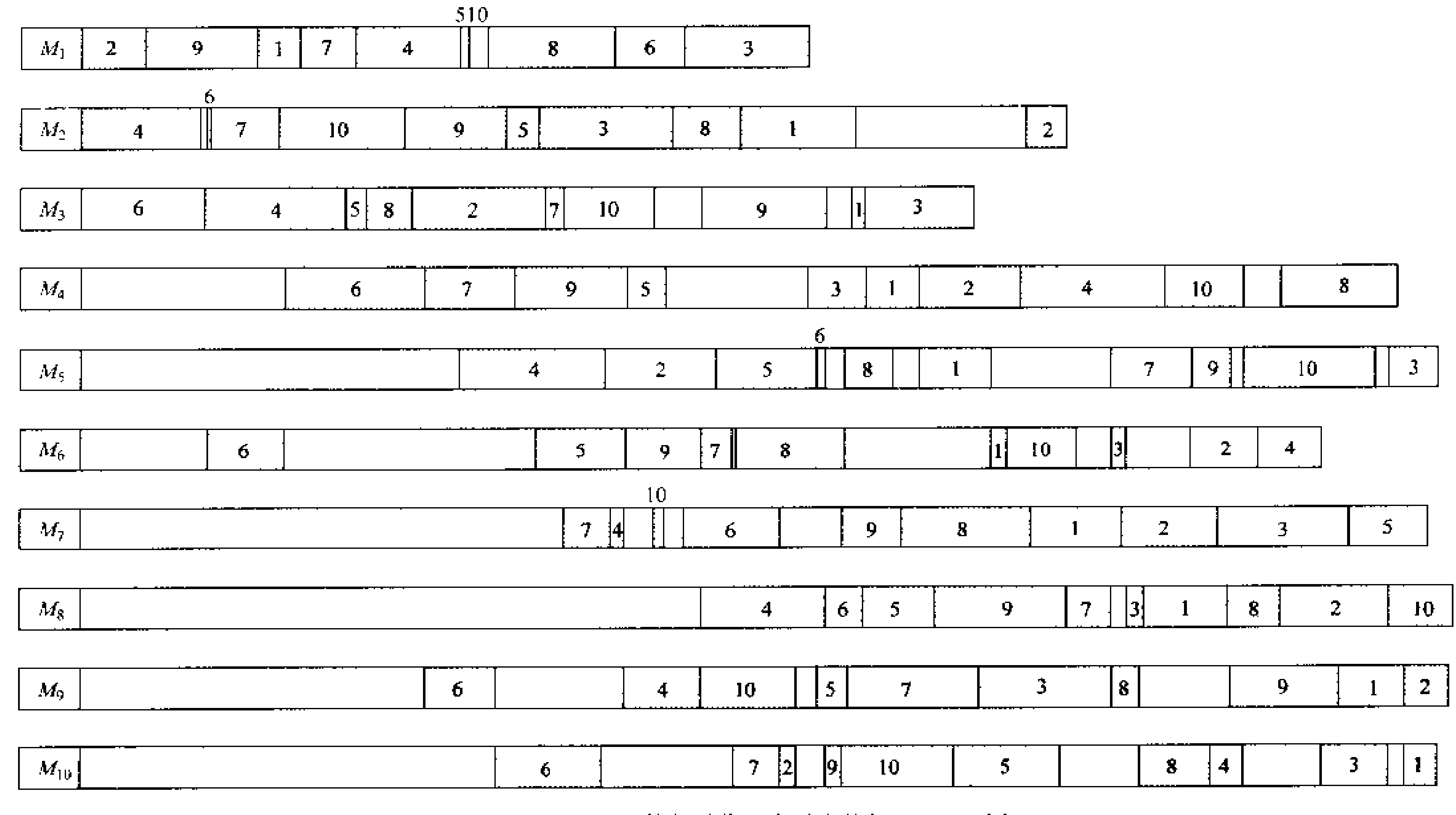


图 3.5.3 FT10 算例最优调度对应的加工 Gantt 图

$$
\boldsymbol{J}_M = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 3 & 5 & 10 & 4 & 2 & 7 & 6 & 8 & 9 \\
2 & 1 & 4 & 3 & 9 & 6 & 8 & 7 & 10 & 5 \\
2 & 3 & 1 & 5 & 7 & 9 & 8 & 4 & 10 & 6 \\
3 & 1 & 2 & 6 & 4 & 5 & 9 & 8 & 10 & 7 \\
3 & 2 & 6 & 4 & 9 & 10 & 1 & 7 & 5 & 8 \\
2 & 1 & 4 & 3 & 7 & 6 & 10 & 9 & 8 & 5 \\
3 & 1 & 2 & 6 & 5 & 7 & 9 & 10 & 8 & 4 \\
1 & 2 & 4 & 6 & 3 & 10 & 7 & 8 & 5 & 9 \\
2 & 1 & 3 & 7 & 9 & 10 & 6 & 4 & 5 & 8
\end{bmatrix}
$$

$$
\boldsymbol{M}_j^* = \begin{bmatrix}
2 & 9 & 1 & 7 & 4 & 5 & 10 & 8 & 6 & 3 \\
4 & 6 & 7 & 10 & 9 & 5 & 3 & 8 & 1 & 2 \\
6 & 4 & 5 & 8 & 2 & 7 & 10 & 9 & 1 & 3 \\
6 & 7 & 9 & 5 & 3 & 1 & 2 & 4 & 10 & 8 \\
4 & 2 & 5 & 6 & 8 & 1 & 7 & 9 & 10 & 3 \\
6 & 5 & 9 & 7 & 8 & 1 & 10 & 3 & 2 & 4 \\
7 & 4 & 10 & 6 & 9 & 8 & 1 & 2 & 3 & 5 \\
4 & 6 & 5 & 9 & 7 & 3 & 1 & 8 & 2 & 10 \\
6 & 4 & 10 & 5 & 7 & 3 & 8 & 9 & 1 & 2 \\
6 & 7 & 2 & 9 & 10 & 5 & 8 & 4 & 3 & 1
\end{bmatrix}
$$

3.6 一类模糊 Job Shop 调度的遗传算法

实际生产调度系统中往往存在大量不确定因素，譬如加工时间的不确定性、工件到达时间的不确定性、交货期的不确定性等，因此研究不确定或模糊加工环境下的车间调度问题很有现实意义，并逐渐受到重视（Slowinski 等，2000）。譬如，Ishii 等（1995）研究提出了模糊优先关系下单机调度问题的数学规划方法、Sakawa 等（1999）提出了模糊加工时间和模糊交货期下 Job Shop 调度的遗传算法，Slowinski 等则专门编著了《模糊调度》一书，但不管怎样，模糊调度问题，尤其是模糊 Job Shop 调度的研究还处于起步和发展阶段，许多方面有待进一步深入研究和完善。下面，简单介绍 Sakawa 等的研究问题及其遗传算法设计。

3.6.1 问题描述和模糊操作

考虑 n 个工件 m 台机器的一般 Job Shop 调度问题，但加工时间和交货期不确

定，其中机器 r 加工工件 j 的第 i 个操作 $o_{j,i,r}$ 的时间由一个三角模糊数(triangular fuzzy number, TFN) $\widetilde{A}_{j,i,r}$ 表征，并用三元组 $(a_{j,i,r}^1, a_{j,i,r}^2, a_{j,i,r}^3)$ 表示(如图 3.6.1 所示)；模糊交货期 $\widetilde{D}_j$ 由相对工件完成时间的满意度来表征，并用二元组 (d_j^1, d_j^2) 表示(如图 3.6.2 所示)。

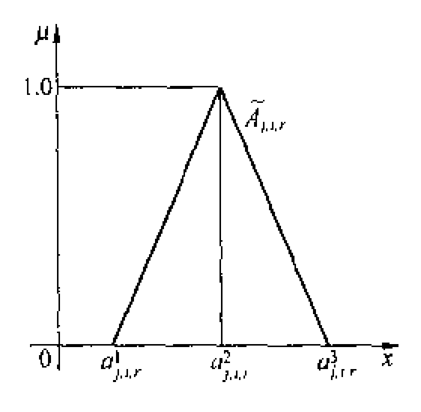


图 3.6.1　模糊加工时间

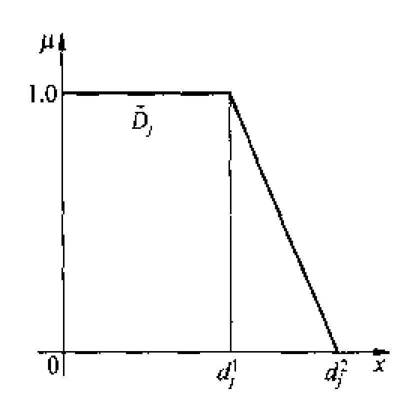


图 3.6.2　模糊交货期

对于每个工件的模糊交货期 $\widetilde{D}_j$，若工件的模糊加工完成时间由 TFN$\widetilde{B}_j$ 表示，则优化目标可以是寻求某个调度方案使得 $\widetilde{B}_j$ 相对 $\widetilde{D}_j$ 的一致指标 AI 的最小值最大。如图 3.6.3 所示，定义 $\widetilde{B}_j$ 相对 $\widetilde{D}_j$ 的一致指标 AI 为两隶属度函数的交叉区域(用阴影区域表示)相对 $\widetilde{B}_j$ 的整个隶属度函数区域的比值，即 AI＝面积 $\widetilde{B}_j \cap \widetilde{D}_j$ 除以面积 $\widetilde{B}_j$，该指标可认为是交货期内完成的 $\widetilde{B}_j$ 的一部分。对于确定性调度问题，模糊完成时间区域变为 0，此时上述一致指标的定义就没有意义，但可将确定加工时间 a 模糊化为 $(a-\delta, a, a+\delta)$，其中 $\delta>0$，然后可用上述定义的极限表示，即 $AI=\lim_{\delta\to 0} AI(\delta)$。

对于这类调度问题，定义如下模糊操作。

1. TFN 的加法操作

累加计算将用于工件的完成时间。若 TFN $\tilde{r}$ 和 $\tilde{t}$ 分别用 (p,r,q) 和 (u,t,v) 表示，则其和 $\tilde{r}+\tilde{t}$ 为 $(p+u, r+t, q+v)$，如图 3.6.4 所示。

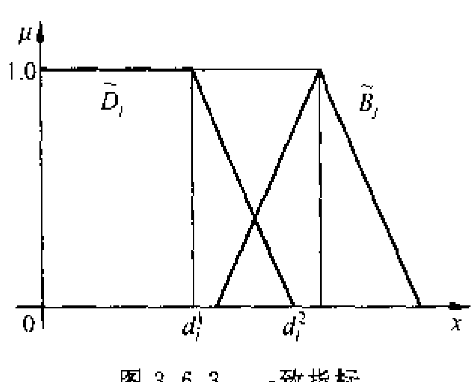


图 3.6.3　一致指标

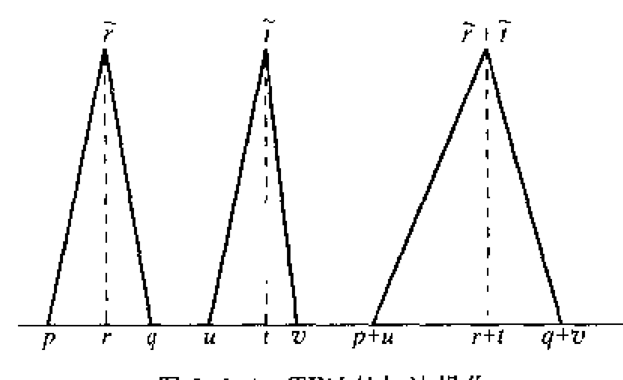


图 3.6.4　TFN 的加法操作

2. TFN的取大操作

取大操作将用于确定工件的加工开始时间。记上述TFN$\tilde{r}$和$\tilde{t}$的隶属度函数为$u_{\tilde{r}}$和$u_{\tilde{t}}$,用$\vee$表示取大操作,则

$$\tilde{r} \vee \tilde{t} = (p,r,q) \vee (u,t,v)$$

其隶属度函数为

$$u_{\tilde{r}\vee\tilde{t}}(z) = \sup_{z=x\vee y}\min\{u_{\tilde{r}}(x), u_{\tilde{t}}(y)\}$$

显然,这种取大操作不能绝对保证模糊数的三角性。由此,在此令

$$(p,r,q) \vee (u,t,v) \approx (p \vee u, r \vee t, q \vee v)$$

来保证三角性。譬如,精确计算意义下,图3.6.5(a)和(b)中,$\tilde{r}\vee\tilde{t}=\tilde{t}$成立,图3.6.5(c)中不成立,但近似操作都成立。例如2×2问题,工件1的加工顺序为机器1—机器2,模糊加工时间分别为(2,5,6)和(5,7,8),工件2的加工顺序为机器2—机器1,模糊加工时间分别为(3,4,7)和(1,2,3),则工件1在机器1和2上的模糊完成时间分别为(2,5,6)和(7,12,14)。由于工件2的第2个操作必须等其第1个操作在机器2上加工完毕以及工件1的第1个操作在机器1上加工完毕后才可进行,因此其加工模糊开始时间为上述两操作完成时间的取大时间,即

$$(3,4,7) \vee (2,5,6) \approx (3,5,7)$$

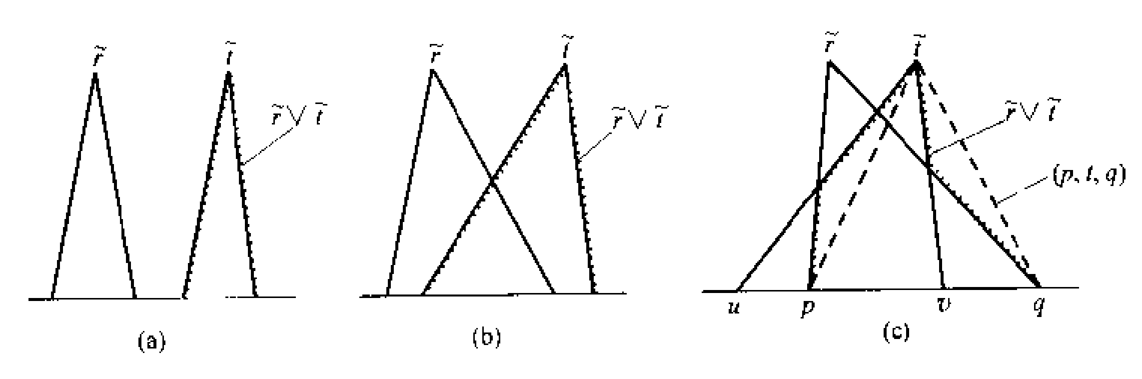


图3.6.5 TFN的取大操作

3.6.2 遗传算法设计

1. 搜索空间

鉴于研究问题的调度指标的正规性,即使加工时间和交货期为模糊数,最优调度仍必定为活动调度,因此可以将搜索空间限于活动调度集。在此,将Giffler-Thompson算法推广到模糊加工时间的情况。

[TFN加工时间下的Giffler-Thompson算法]:

步骤1:令集合C为所有工件第1道操作的结合。

步骤2:假定机器在同一时刻可以加工任意多个工件,计算C中各操作$o_{j,i,r}$的模糊完成时间,记作$(EC^1_{j,i,r}, EC^2_{j,i,r}, EC^3_{j,i,r})$。

步骤 3：确定 C 中 EC^1 最小的操作 o_{j^*,i^*,r^*}，将 C 中所有将在机器 r^* 上加工的操作 o_{j,i,r^*} 构成冲突集合 G。

步骤 4：从 G 中随机选择一个操作 o_{j_s,i_s,r^*}，以其为基准通过模糊取大（对加工开始时间的调整）和模糊加法操作（对加工完成时间的调整）更新冲突集合中其他操作的 EC^1，EC^2，EC^3 值（因为事实上每台机器同一时刻只能加工一个工件，所以必须对步骤 2 所确定的模糊时间作调整），最后在 C 中移去操作 o_{j_s,i_s,r^*}，而将其下道工序加入 C，并计算其相应的模糊完成时间。

步骤 5：重复步骤 3 至 4，直到所有操作加工完毕。

显然，上述算法可用于产生模糊加工时间下的活动调度，其中操作 o_{j^*,i^*,r^*} 的确定也可以采用其他指标。

2. 编码

染色体用模糊完成时间的 $3n\times m$ 维矩阵表示，它表征了 n 个工件 m 台机器的一个调度。显然，由于加工数据的模糊性，这种编码与前文介绍的确定性 JSP 问题的遗传算法编码很不相同，不仅复杂，而且存储量较大，且必须设计特殊的遗传操作。

3. 种群初始化与相似度计算

鉴于 TFN 加工时间下的 Giffler-Thompson 算法的步骤 4 的随机性，多次运行可产生不同的活动调度。为了保持初始种群的多样性，要求初始种群的相似度不能够大于某个设定值（譬如 0.8），其中相似度的定义由以下 4×4 示例来解释。假定两个个体对应的加工顺序如表 3.6.1 所列，对比这两个个体相同机器上的工件顺序可见，机器 1 上与工件 1 加工先后关系固定有工件 3 和 4，因此赋值 2，机器 1 上与工件 2 先后关系固定的也是工件 3 和 4，因此赋值 2，机器 1 上与工件 3 先后关系固定的有工件 1、2 和 4，因此赋值 3，机器 1 上与工件 4 先后关系固定的有工件 1、2 和 3，因此赋值 3，由此对机器 1 赋值 2＋2＋3＋3＝10；同理，对机器 2 赋值 3＋3＋3＋3＝12；对机器3 赋值 3＋3＋3＋3＝12；对机器 4 赋值 3＋3＋2＋2＝10。因此，相似度为（10＋12＋12＋10）/48＝0.917。

表 3.6.1　个体 1 和 2 对应的加工顺序

个体 1					个体 2				
机器	工件				机器	工件			
1	1	2	4	3	1	2	1	4	3
2	3	1	4	2	2	3	1	4	2
3	4	3	1	2	3	4	3	1	2
4	3	2	1	4	4	3	2	4	1

4. 遗传操作(交叉、选择和变异)

活动调度生成算法的关键是对冲突操作的处理,也就是步骤 4。对于所选父代个体,一旦一个冲突发生,就有一半概率使加工顺序朝消除冲突的方向进行,由此可生成后代个体,而重复 k 次将得到 k 个后代。进而,在该 k 个后代个体和 2 个父代个体中保留最小一致指标值最大的两个个体,也即局部排名选择。显然,采用这种选择操作无需定义适配值函数,同时,k 值越大,得到好个体的概率越大。但由于它们由同一个体产生,故后代个体的相似度将增加。至于变异操作,在进行上述交叉操作时,冲突操作不是从父代个体中选择,而是从冲突集合 G 中以变异概率选取。

5. 种群构造

为了防止收敛到局部解,遗传算法中采用了一种特殊的种群构造方式。如图 3.6.6所示,假设有 3 组种群,每组基于相似度产生 10 个个体。当各组种群均收敛到某一个度时,将各组种群合并继续进行进化直到收敛。

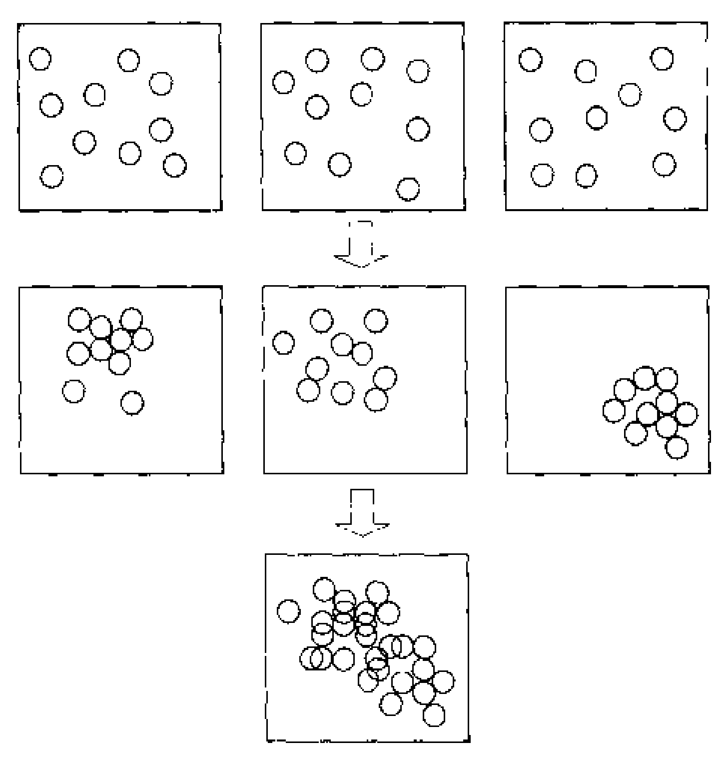


图 3.6.6 种群构造

3.6.3 仿真结果

考虑如下 6×6 问题(其加工数据如表 3.6.2 所列),以及 10×10 问题(其加工数据如表 3.6.3 所列)。

表 3.6.2　6×6 问题的加工数据

工件号	机器号及顺序(加工时间)						交货期
1	1(5,6,13)	3(3,4,5)	2(1,2,3)	6(3,4,5)	4(2,3,4)	3(2,3,4)	30, 40
2	1(3,4,5)	2(2,4,5)	3(1,3,5)	6(4,5,6)	4(5,6,7)	5(6,7,8)	35, 40
3	3(1,2,3)	6(5,6,7)	5(4,5,6)	4(3,4,5)	2(1,2,3)	1(1,2,3)	20, 28
4	6(2,3,4)	5(1,2,3)	4(2,3,4)	2(2,3,5)	1(3,4,6)	3(3,4,5)	32, 40
5	6(3,4,5)	5(2,3,4)	4(1,2,3)	3(2,3,4)	2(4,5,6)	1(2,3,4)	30, 35
6	5(6,7,8)	6(4,5,6)	1(2,3,4)	2(3,4,5)	3(2,3,4)	4(1,2,3)	40, 45

表 3.6.3　10×10 问题的加工数据

工件号	机器号及顺序(加工时间)										交货期
1	8(2,3,4)	6(3,5,6)	5(2,4,5)	2(4,5,6)	1(1,2,3)	3(3,5,6)	9(2,3,5)	4(1,2,3)	7(3,1,5)	10(2,3,4)	45,60
2	10(2,3,4)	7(2,3,5)	4(2,4,5)	6(1,2,3)	8(4,5,6)	3(2,4,6)	2(2,3,4)	1(1,3,4)	5(2,3,4)	9(3,4,5)	50,60
3	6(2,4,5)	9(1,2,3)	10(2,3,5)	8(1,2,4)	1(3,5,6)	7(1,3,4)	4(1,3,5)	2(1,2,4)	5(2,4,5)	3(1,3,5)	50,65
4	1(1,2,3)	5(3,4,5)	8(1,3,5)	9(2,4,6)	10(2,4,5)	6(1,2,4)	7(3,4,5)	2(1,3,5)	4(1,3,6)	3(1,3,4)	50,65
5	2(2,3,4)	7(1,3,4)	3(1,3,4)	5(1,2,3)	8(1,3,5)	9(2,3,4)	10(3,4,5)	6(1,3,4)	1(3,4,5)	4(1,3,4)	50,65
6	4(2,3,4)	2(2,3,4)	3(1,2,3)	5(2,4,5)	6(1,3,4)	8(1,3,4)	7(3,4,5)	9(1,2,3)	10(2,4,5)	1(1,3,4)	45,60
7	3(2,3,4)	5(1,4,5)	4(1,3,5)	1(3,4,5)	9(2,3,4)	7(3,4,5)	2(1,2,3)	10(3,5,6)	8(3,5,6)	6(1,2,3)	45,60
8	7(3,4,5)	1(1,2,3)	9(3,4,5)	6(2,4,5)	10(1,3,4)	2(2,3,4)	5(1,2,3)	3(2,4,5)	4(3,4,5)	8(2,3,5)	50,60
9	9(3,4,5)	4(1,3,4)	10(1,3,5)	2(2,3,4)	3(3,5,6)	5(2,4,5)	8(1,3,4)	1(5,4,5)	5(1,2,3)	7(3,4,5)	15,60
10	7(2,4,5)	5(1,2,3)	2(3,4,5)	4(2,3,4)	1(1,2,3)	8(3,1,5)	10(2,4,5)	6(3,4,5)	3(1,2,3)	9(1,2,4)	50,60

遗传算法中 $k=3$,变异概率为 5%,求解 6×6 问题的种群大小为 30,进化代数为 50,求解 10×10 问题的种群大小为 50、进化代数为 80。对每个算例均随机运行 10 次仿真,遗传算法得到的结果如表 3.6.4 所列。

表 3.6.4　遗传算法的优化结果

实验次数		1	2	3	4	5	6	7	8	9	10
最小一致指标	6×6 问题	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69
	10×10 问题	0.94	0.94	0.94	0.94	0.94	0.93	0.94	0.94	0.93	0.93
平均 CPU 时间*	6×6 问题	17.6s									
	10×10 问题	170.4s									

* Fujitsu S-4/10 工作站

可见,上述遗传算法能够较快速、高质量地解决模糊加工时间和模糊交货期下的 JSP 问题。需要强调的是,遗传算法的种群并行进化搜索,尤其在增加相似度计算和

种群构造环节情况下，算法实现最优化的成功率大大增加。

3.7 Job Shop 调度的遗传算法简要综述

近年来，随着先进制造技术和计算机技术的发展，JSP 调度的含义有所推广，主要体现在问题的随机性、动态性、不确定性、约束性、多目标等方面，从而与实际生产更为接近，但开发求解 JSP 的有效算法仍一直是调度和优化领域的重要课题。如何利用 GA 高效求解 JSP 也一直被认为是一个有挑战意义的难题并成为一个研究热点。近年来，围绕该方面研究涌现出大量论文并取得较大进展，在此对基于 GA 的 JSP 研究进行简要介绍。

3.7.1 JSP 的 GA 编码研究

JSP 的 GA 编码技术参见本书 3.3 节，在此不再赘述。需要指出，编码是 GA 应用能否成功的关键，它一直是 GA 求解 JSP 的瓶颈。有针对性地设计有效的编码方式，并基此开发高效的遗传操作，无疑有助于 GA 搜索效率与质量的提高。

3.7.2 JSP 和 GA 的特征分析

至于特征分析，一方面可通过分析 JSP 的特征来使其适应 GA 的优化过程，另一方面则可通过分析 GA 的特征来使其适应 JSP 的特点。当然，两者也可结合使得 GA 和 JSP 互相适应，所有这些研究均有助于利用 GA 更好地解决 JSP 问题。

JSP 调度不仅具有 NP-Complete 特性，同时又是一个受约束系统，且其约束具有很强的关联性，分析 JSP 的特征可得到指导搜索的有效启发式知识或规则，进而在结合基于知识的系统下构成知识驱动的调度系统。段黎明等(1998)对基于约束分析的 JSP 调度算法进行了综述，包括一致性增强方案、变量排序启发式和值排序启发式算法、向后看方案中的主要方法，而基于修改的约束分析方法则是有待发展的方法。Jain 等(1999)通过对大量 Benchmark 问题的分析，基于工件和机器数的关系提出区分 JSP 难易的准则，进而有助于工程人员采用不同的方法或态度来对待特定问题。陈恩红等(1998)分析了 JSP 的特点，称各种工件的生产顺序为需求空间，生产每种工件可采用的不同规划实现为规划空间，每种规划中的每种操作在不同机器上的实现为资源空间，进而给出了面向资源空间与面向规划空间的遗传操作设计。纪树新等(1997)通过分析 JSP 的数学表达模型和特点，研究了 JSP 基因的若干概念，并详细论述了连锁基因编码法的机理和可行性。Wang 等(2000)定量分析了 JSP 的不可行解的结构，并给出了 JSP 发生死锁的一个充分必要条件，同时对两机问题提出了一个不可行解的计算公式，这对启发式方法的构造具有一定的指导意义。此外，研究调度特性(如活动调度、半活动调度、非迟延调度等)与性能指标(如 Makespan

等正规性能指标、E/T 等非正规性能指标)的关系,有助于算法编码和操作的设计。譬如基于正规性能指标的最优解必然是活动调度,若能将搜索空间限定在活动调度集上,而不是范围更广的半活动调度集,则既能保证搜索状态的可行性,又有助于提高搜索效率,并保证搜索到最优解的可能。

虽然近年来 GA 应用于 JSP 的研究很多,但大都局限于少数几个问题,因此所得到的关于 GA 求解这类问题的优化性能的结论是否具有普遍性呢? Hart 等(2000)为了深入调查 GA 对 JSP 这类问题的优化性能,进而分析 GA 的优势与弱点,设计了一个可调整难度的可配置问题发生器,其研究表明 GA 具有相当好的鲁棒性,即 GA 能够在绝大多数情况下得到大部分问题的理想解,并且有能力在 3 到 4 次实验中得到最优解。这个结论显然是实际工程领域所希望的,从而也使得 GA 更具吸引力。但不可否认,其前提是算法必须得到很好的设计,从而又涉及到许多 GA 的理论与实现问题的研究,这不仅需要对 GA 进行更深入地研究,而且也需要我们对 GA 更深入地理解。

此外,鉴于 GA 的自然选择和进化操作机制,若干代进化后的生存者必然具有相似的特性。就 JSP 而言,GA 得到的优良个体在操作和排序的特征方面存在相似的联系,通过探索这种属性来构造规则将能模拟 GA 的行为性能,进而利用所构造的规则即可求解类似的问题。Koonce 等(2000)利用数据挖掘算法从由 GA 生成的调度集中抽取知识,从而开发一个规则集调度器来逼近 GA 的行为,在求解类似问题时其性能普遍优于一般的简单分配规则。

3.7.3 Benchmark 问题和算法改进与比较研究

JSP 的 Benchmark 问题参见本书 3.2 节,在此不再赘述。

尽管 GA 具有种群并行搜索能力,在总体上把握了进化的方向,但其局部搜索能力较差,容易出现早熟现象。对 GA 进行改进主要可表现在算法的编码、初始化、操作、算法结构、优化机制等方面。Cheng 等(1999)综述了求解 JSP 的交叉和变异操作的设计。Dellacroce 等(1995)提出了一种基于优先规则的编码的遗传算法。Dorndorf 等(1995)用 GA 进化工件分配的优先规则序列和 SB(Shifting Bottleneck)方法意义下的单机解序列。Shi(1997)针对基于操作的编码,提出了有效的交叉方式,在保证可行性的基础上取得了较好的性能。Ombuki 等(1998)通过在遗传码中嵌入启发式方法的知识提出了一种 gkGA 方法,从而启发式规则在优化过程中同样得到优胜劣汰。Varela 等(1999)构造了一种利用概率方法提供的特定知识信息来产生遗传算法初始种群的启发式方法,进而改进了算法的优化性能。Wang 等(2000)提出了一种简单而通用的编码方式和相应的遗传操作,并在算法中嵌入启发式规则来改善算法质量和效率。Hajri 等(2000)提出了一种受控遗传算法,他们采用并行机编码,用启发式方法产生初始种群,并利用了多交叉操作,同时基于模糊逻

辑和置信函数来估计遗传操作的参数。Tezuka 等(2000)基于一种新的编码,提出了一种有效的 CCX 交叉(common cluster crossover)来改进问题的设置时间。Qi 等(2000)提出了一种并行多种群的遗传算法来满足 JSP 的动态特性。

同时,近年来许多学者在提出一些新的或改进方法的同时,往往与一些现有方法进行比较。算法比较不仅有助于验证其性能,而且有助于对算法的行为特点有更感性的认识,为更好地应用或改进算法提供条件。Tadei 等(1995)比较了 SB,TS,SA,GA 和 Lagrangian 松弛法等解决 JSP 的先进搜索方法的性能。Kurbel 等(1995)将 SA,GA 和混合整数线性优化应用于 JSP,并对两者的优化和时间性能作了比较。Blazewicz 等(1996)详细介绍和比较了 JSP 的一些传统和近期的求解方法。Ponnambalam 等(1999, 2000)研究了 SA,TS 的求解性能,并与已有的 GA 等结果进行了比较。

3.7.4 混合遗传算法的研究

算法混合的思想是提高 GA 搜索效率和质量的有效手段,如在 GA 中嵌入局部搜索方法则能改善其局部搜索能力,嵌入基于知识的启发式方法则能够将一些问题相关的信息融入到 GA 的全局搜索中,进而增强 GA 的整体的搜索潜力。王凌等(2001)较全面地讨论了混合算法的结构和设计问题,而求解 JSP 的混合遗传算法主要表现在为产生初始种群的混合、为评价个体适配值的混合、为增强局部搜索能力和克服早熟收敛的混合等方面。

Dagli 等(1993)针对非线性多准则目标函数的离散调度环境提出了 GA 和人工神经网络的混合算法,其中 GA 用于从一组随机可行调度搜索得到最优调度,而经训练后的神经网络则作为一个多准则评价器通过建立由调度准则集合到由"有经验的专家"提供的评价值之间的映射模型来得到个体的适配值。Lee 等(1997)结合 GA 和一种称为诱导决策树的机器学习技术的互补能力,开发了一个 JSP 调度系统,这种利用 GA 来分配各机器上的工件以及利用机器学习技术来释放工件到车间层的做法对动态调度问题是很有前景的。顾擎明等(1998)建立 JSP 的神经网络模型,网络的平衡态对应于问题的有效解,它使优化过程始终保持在较快到达最优的方向上进行,而 GA 则以种群为单位向整个解空间扩展,从而保证混合算法能够快速找到稳定优化解。杨圣祥等(1998)用 GA 结合基于约束满足的自适应神经网络求解 JSP,其中 GA 用于迭代优化,当交叉和变异生成的个体为不可行解时,由自适应神经网络运算后得到可行解,从而生成新一代种群。Kolonko(1999)将小种群 SA 运行于 GA 框架,利用自适应温控和重升温以及面向时间的交叉来解决 JSP 问题。Mesghouni 等(1999)结合 GA、约束逻辑编程(CLP)和多准则决策(MCDM)来解决 JSP 问题,其中基于运筹学和人工智能的 CLP 为 GA 提供一组包含问题约束的初始解,GA 则用于得到一组优良调度,MCDM 则基于多准则从中选择最满意的调度。李小平等(1999)

提出了一种定界-遗传算法。他们首先用一个定界算法排出一个初始加工顺序,并以其目标值为界随机产生用于GA进化的初始种群,如此减小了GA在低适配值种群中搜索的盲目性。Cheng等(1999)介绍了一些求解JSP的混合算法,包括自适应遗传算子、基于启发式特征的遗传算子和混合遗传算法的设计。Cai等(2000)结合局部搜索方法和GA有效求解JSP问题。Wu等(2000)提出了GA和成组技术的混合方法,成组技术的应用降低了问题的复杂性,JSP在一定程度上转化为Flow Shop问题。姚伟力等(2000)在求解E/T指标下的JSP时,先GA用于确定可行调度序列,然后用模糊控制器对开工时间加以调整。Wang等(2001)结合GA和SA提出了一类混合优化策略,通过对Benchmark问题的研究验证了混合算法较单一算法的优越性。

近年来,混合遗传算法的研究在国际上受到广泛重视。Davis(1991)对GA提出了"hybridize where possible",Dimopoulos等(2000)则认为混合算法的研究无疑是一个新的发展趋向。

3.7.5 JSP的推广和动态调度

过于简化的调度模型虽然便于理论研究,但却难以用于复杂的实际生产过程。因此,如果JSP的研究仅局限在理想的数学模型,则大量的具有建模困难、计算复杂、存在动态性、随机性、约束性以及多目标等难点的实际调度问题仍无法得到解决。生产过程的随机性、不确定性往往要求不断进行重新调度,这不仅需要处理各种突发事件来适应各种复杂多变的加工环境,而且还要避免因追求全局优化带来的巨大计算量,因此近年来JSP的研究延伸到更具实际意义的动态调度和更具实际特性的VLSI、装配线、间歇调度等问题。

方剑等(1997)借助预测控制中滚动优化的思想提出了周期性和事件驱动的滚动调度策略,将GA和分配规则相结合来处理同操作序列有关的工件安装时间和工件到期时间约束的复杂调度问题。Lee等(1997)研究了GA在批量可变的柔性流水线加工问题的应用,他们将工件的批量和排序同时加以优化,并与SA,TS等方法加以比较。Ozdamar等(1998)研究了一类具有设置时间和超时决策的CLSLP(capacitated lot sizing and loading problem)问题,并提出了结合SA,GA和TS的混合启发式方法。翁妙凤(2000)针对具有路径柔性的动态JSP提出了基于进化规划、周期性和事件驱动的两种有(无)窗口的重调度策略。Prins(2000)针对一类具有自由工件路径的Open-Shop问题,给出了几种相应的GA解决方案。钱晓龙等(2001)则对动态调度的研究方法进行了综述。

同时,鉴于丰田公司JIT技术的成功应用,基于交货期的E/T调度成为一个新的JSP研究热点。童刚等(2000)讨论了不同交货期窗口下的E/T并行机调度问题,并给出了相应的稳步遗传算法。顾擎明等(1998)研究了基于交货期要求、优先级要

求和有限资源约束下 JSP 的自适应遗传算法。Sakawa 等(2001)讨论了双目标下的具有模糊加工时间和模糊交货期的模糊 JSP,并提出了结合模糊技术和 GA 的有效方案。

此外,间歇化工过程中的 Multi purpose 问题是 JSP 的一种扩展,它除包含 JSP 的组合特性外,还需确定一些受约束的连续变量,其优化过程更为复杂,将解决 JSP 的方法继承到 Multi-purpose 问题显然有助于间歇化工调度的发展(王凌等, 2000)。

3.7.6 调度器开发和实际应用

理论研究必须面向应用并得到实践,基于 GA 的 JSP 研究同样不能仅仅局限于理论和学术论文,将理论成果转化为实用技术并加以实际应用是十分有价值的工作。

开发基于 GA 的仿真系统和调度器近年来取得较大进展。Maturana 等(1997)提出了一个集成交货期算法、GA 和生产单元仿真器的面向对象的调度系统。该系统各模块通过交互信息来完成遗传优化并同时满足时间依赖和技术约束,其中交货期算法利用工件的产品顺序和操作时间来生产产品优先权,GA 产生可行机器路径和调度的一个种群并传给生产单元仿真器,后者在不同的生产组态下履行生产计划。Kim 等(1998)提出了遗传增强学习(GRL)方法,把增强学习的策略加入遗传编码,进而开发了基于 GRL 的 EVIS(evolutionary intracell scheduler)调度器,并用于多种类型调度问题。鉴于手工建立 JSP 的仿真模型十分繁琐,王宇等(1999)给出了一种基于 SIMAN/CINEMA 软件的动画仿真程序自动生成系统,用以方便地生成标准 JSP 的动画仿真模型程序,并可推广到生产管理、交通运输、邮电等领域。Cardon 等(2000)将 GA 和 Multi-agent 技术在 JSP 中加以应用,提出了一个基于合同网协议(contract-net protocol)的动态模型。Khoo 等(2000)开发了一个生产系统的 GA-enhanced多目标调度器原型,该原型包括一个调度模型构成的调度工具箱,系统通过接受数据库或文件来的输入数据则可产生次优的调度方案。

同时,GA 在许多实际工程领域得到了应用,并取得了认可。Candido 等(1998)将 GA 应用于一系列实际 JSP 问题,并与局部爬山法相结合。Hart 等(1999)将 GA 用于一个地方鸡肉加工厂。其研究表明,对于这个具有强约束的实际问题,GA 能够在几分钟内有效地产生日调度。Baudet 等(1999)将 GA 和离散事件仿真模型相结合用于处理不连续精细化工厂的短期调度。相关的研究还有很多,在此不一一列举。

3.7.7 展望

可以说,JSP 调度是生产调度领域的典型代表,GA 则是智能优化技术的代表,因此基于 GA 的 JSP 研究具有重要的理论意义和工程价值。通过对编码、特征分析、算法改进、比较、混合算法、推广问题、调度器开发和实际应用等有关研究的介绍,该问题的重要性和关注程度可见一斑。然而,大量的工作还有待进一步深入和完善。

首先,"Why do metaheuristics work as well as they do?"至今仍是有关GA等Metaheuristics的一个重要的开放性理论问题(Hansen等,2001)。结合研究JSP的特性及其局部极小拓扑结构的分析和GA遗传算子的特性分析与结构设计,有助于更好利用GA解决JSP问题,而这方面的研究目前还很少,很不深入,因此对于具有很强工程背景的JSP这个代表性问题,这无疑是一个很重要的研究方向。同时,进一步开展GA的理论研究(如编码、参数选取、操作设计、收敛速度估计、算法比较等),并利用数据挖掘、智能体等先进技术进行分析和研究,都能使我们更深入地认识GA,并推动其体系的发展,拓宽其应用的领域。

其次,将JSP的数学模型更加现实化,动态调度的研究尤其值得重视,并将已有的理论成果推广到实际工程领域,用实践来检验技术的潜力,这不仅能够推动先进生产的发展,而且将促使优化技术的进一步完善。同时,鉴于GA对生产技术人员较为陌生,开发相关的实用化集成化调度平台和优化技术,无疑具有很强的工程意义。

再则,鉴于国际学术界对混合系统与算法的日益重视,混合优化策略以及相应仿真环境的开发无疑是一个值得关注的研究方向,尤其是将已有各种调度方法的优点与GA有效地融和。

总之,随着计算技术、优化技术、数学理论和计算机硬件的发展,GA、JSP本身以及它们的结合研究将得到更完善的解决,研究与应用领域将更宽广,理论研究将更系统化,更接近于实际,并朝集成化、实用化、最优化、多目标化、动态化、规模化发展。

第4章　Flow Shop 调度及其遗传算法

本章首先介绍 Flow Shop 调度的描述、常用启发式方法和若干 Benchmark 问题及其相关的研究结果，然后分别介绍置换 Flow Shop 调度的一类改进遗传算法、多目标 Flow Shop 调度的遗传算法、一类批量可变的 Flow Shop 调度的遗传算法、模糊 Flow Shop 调度的遗传算法，最后介绍混合 Flow Shop 调度的遗传算法。

4.1　引言

4.1.1　问题描述

Flow Shop 调度问题（flow shop scheduling problem，FSP）是许多实际流水线生产调度问题的简化模型，也是一个典型的 NP-hard 问题，因此其研究具有重要的理论意义和工程价值，它也是目前研究最广泛的一类典型调度问题。Flow Shop 调度研究 m 台机器上 n 个工件的流水加工过程，每个零件在各机器上加工顺序相同，同时约定每个工件在每台机器上只加工一次，每台机器一次在某一时刻只能够加工一个工件，各工件在各机器上所需的加工时间和准备时间已知，要求得到某调度方案使得某项指标最优。进一步，若约定每台机器加工的各工件的顺序相同，则称其为置换 Flow Shop 问题。

置换 Flow Shop 调度问题的数学描述如下。令 t_{ij} 为工件 i 在机器 j 上的加工时间，θ_{ij}^{k} 为机器 k 上加工完工件 i 后马上加工工件 j 所需的准备时间（若不加特殊说明，$\theta_{ij}^{k}=0$），T_i 为工件 i 的加工完毕时间，D_i 为工件 i 的计划完成时间，不失一般性，假设各工件按机器 1 至 m 的顺序进行加工，令 $\pi=(\sigma_1,\sigma_2,\cdots,\sigma_n)$ 为所有工件的一个排序，则

$$\begin{cases} T_{\sigma_1,1}=t_{\sigma_1,1} \\ T_{\sigma_j,1}=T_{\sigma_{j-1},1}+\theta_{\sigma_{j-1},\sigma_j}^{1}+t_{\sigma_j,1}, \quad j=2,\cdots,n \\ T_{\sigma_1,i}=T_{\sigma_1,i-1}+t_{\sigma_1,i}, \quad i=2,\cdots,m \\ T_{\sigma_j,i}=\max\{T_{\sigma_{j-1},i}+\theta_{\sigma_{j-1},\sigma_j}^{i},\ T_{\sigma_j,i-1}\}+t_{\sigma_j,i}, \quad i=2,\cdots,m;\ j=2,\cdots,n \end{cases} \tag{4-1-1}$$

$$\pi^{*}=\arg\{f(\pi)=\sum_{i=1}^{n}\max(T_{\sigma_i}-D_{\sigma_i}),\quad 0\}\rightarrow\min \tag{4-1-2}$$

或

$$\pi^{*}=\arg\{f(\pi)=T_{\sigma_n,m}\}\rightarrow\min \tag{4-1-3}$$

其中,式(4-1-2)表示以最小拖延时间为指标,而式(4-1-3)则表示以最大完成时间为指标,即 Makespan 指标。

对于一般 Flow Shop 调度问题,每台机器加工各工件的顺序可不同。不失一般性,假设各工件按机器 1 至 m 的顺序进行加工,令工件加工顺序矩阵为 $S=(s_{i,j})_{m\times r}$,其中 $s_{i,j}$ 表示机器 i 上第 j 个加工的工件号码,则

$$\begin{cases} T_{s_{1,1},1}=t_{s_{1,1},1} \\ T_{s_{1,i},1}=T_{s_{1,i-1},1}+\theta^1_{s_{1,i-1}s_{1,i}}+t_{s_{1,i},1}, \quad i=2,\cdots,n \\ T_{s_{j,1},j}=T_{s_{j,1},j-1}+t_{s_{j,1},j}, \quad j=2,\cdots,m \\ T_{s_{j,i},j}=\max\{T_{s_{j,i-1},j}+\theta^j_{s_{j,i-1}s_{j,i}},\ T_{s_{j,i},j-1}\}+t_{s_{j,i},j}, \quad i=2,\cdots,n;\ j=2,\cdots,m \end{cases} \tag{4-1-4}$$

$$S^*=\arg\{f(S)\}=\sum_{i=1}^{n}\max\{(T_{s_{m,i},m}-D_{s_{m,i}}),\quad 0\}\rightarrow\min \tag{4-1-5}$$

或

$$S^*=\arg\{f(S)=T_{s_{m,n},m}\}\rightarrow\min \tag{4-1-6}$$

其中,式(4-1-5)和式(4-1-6)的意义分别同于式(4-1-2)和式(4-1-3)。

尽管相对 Job Shop 调度而言,Flow Shop 的工艺约束比较简单,但它仍旧是一个非常复杂和困难的组合优化问题,其 NP-hard 特性和强大的工程背景使其一直成为理论界和工程领域研究的热点问题。

4.1.2 启发式方法

Flow Shop 调度问题的求解方法通常可分为精确方法、构造型方法、改进型方法和神经网络等。精确方法,如列举法、分支定界、动态规划等,计算量和存储量大,仅适合于小规模问题。构造型方法通过一定的规则来构造问题的解,如 Gupta 法、Johnson 法、Palmer 法、CDS 法、RA 法、NEH 法、SL 法、WSH 法等,这类算法能快速构造解,但通常解的质量较差。改进型算法从若干解出发,通过对其邻域的不断搜索和当前解的替换来改进解的质量,如遗传算法、模拟退火、进化规划、禁忌搜索等,它们通常可取得较好的优化效果,但往往需要大量迭代,且算法操作、参数和结构需要合适选取。自 Hopfield 用神经网络求解 TSP 问题之后,神经网络也成为求解调度问题的有效方法,它通过神经网络的动态演化来达到对应优化解的稳定态,但网络参数和能量函数要精心设计,并且算法复杂性较大。

研究表明,3 台以上机器的 Flow Shop 调度即为 NP-hard 问题,至今没有一个多项式复杂性的全局优化算法。因此,为了快速得到问题的次优解,许多工程领域和调度研究人员设计了若干启发式方法。下面简单介绍几种求解置换 Flow Shop 调度的启发式方法,不加特殊说明,调度指标为 Makespan。

(1) Palmer 方法

记 t_{ij} 为工件 i 在机器 j 上的加工时间，该方法首先对每个工件计算参量 $S(i) = \sum_{j=1}^{m}(2j-m-1)t_{ij}/2, i=1,\cdots,n;\ j=1,\cdots,m$，然后将工件按参量 $S(i)$ 值递减的顺序进行排列，从而得到一个次优调度。

(2) Gupta 方法

该方法首先对每个工件计算参量 $S(i)=C/\min\limits_{1\leqslant j\leqslant m-1}(t_{ij}+t_{i(j+1)})$，其中若 $t_{im}\leqslant t_{i1}$ 则 $C=1$，否则 $C=-1$，然后将工件按参量 $S(i)$ 值递增的顺序进行排列，从而得到一个次优调度。

(3) Bonney-Gundry(BG)方法

该方法假设工件一旦开始加工就不允许在加工中途间断工序，直到所有工序的加工完成，即 No-wait 加工。图 4.1.1 描绘了工件 1 在 3 台机器上的不间断加工过程。同时，该方法采用线性回归方法，对每个工件计算其加工时间的开始斜率和结束斜率。进而，将开始斜率最大的工件排列在第 1 个位置，然后将其余工件中开始斜率最接近前一个工件结束斜率的工件排列在下一个位置，依次类推，从而得到一个次优调度。

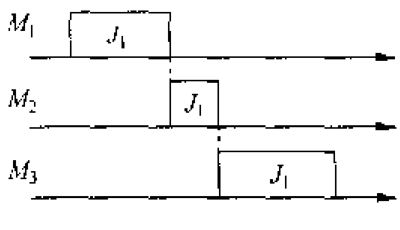


图 4.1.1 工序不间断加工过程

(4) Campleu-Dudek-Smith(CDS)方法

该方法首先计算参量 $T_{i1}(r)=\sum_{j=1}^{r}t_{ij}$ 和参量 $T_{i2}(r)=\sum_{j=m-r+1}^{m}t_{ij}$，然后对 r 从 1 至 $m-1$ 采用 Johnson 方法分别求解得到 $m-1$ 个排序，进而采用其中目标值最小的排序为次优调度。其中，双机调度的 Johnson 方法对于双机问题能够给出最优调度，其步骤如下：

[$n/2/F/F_{\max}$调度的 Johnson 方法]：

步骤 1：令 $k=1, l=n$。

步骤 2：令当前未排序表为 $\{J_1, J_2, \cdots, J_n\}$。

步骤 3：对未排序的工件，找出在一台机器上的最短加工时间和相应的工件编号。若最短加工时间出现在第 1 台机器上，则转步骤 4；否则转步骤 5。

步骤 4：对最短加工时间出现在第 1 台机器上且相应的工件为 J_i，依次进行如下操作：

(4.1) 将工件 J_i 排列在加工工序的第 k 位，然后从未加工工件表中去掉该工件。

(4.2) 令 $k=k+1$，然后转步骤 6。

步骤 5：对最短加工时间出现在第 2 台机器上且相应的工件为 J_i，依次进行如下操作：

(5.1) 将工件 J_i 排列在加工工序的第 l 位，然后从未加工工件表中去掉该工件。

(5.2) 令 $l=l-1$，然后转步骤 6。

步骤 6：构成去掉已排序工件后的新未排序工件表。若该表非空则转步骤 3，否则结束算法。

(5) Dannenbring 方法

该方法首先计算参量 $T_{i1}=\sum_{j=1}^{m}(m-j+1)t_{ij}$ 和参量 $T_{i2}=\sum_{j=1}^{m}jt_{ij}$，然后把原多机排序问题转化为以 T_{i1} 和 T_{i2} 为加工时间的双机调度问题，进而用 Johnson 方法得到一个次优调度。

(6) Nawaz-Enscore-Ham(NEH)方法

该方法的基本思想是赋予总加工时间越长的工件越大的在排列中插入优先权，即首先计算各工件在所有机器上的加工时间和，并按递减顺序排列，然后将前两个工件进行最优调度，进而依次将剩余工件逐一插入到已调度的工件排列中的某个位置，使得调度指标最小，直到所有工件调度完毕，从而得到一个次优调度。

(7) Rajendran 方法

该方法对 NEH 的工件排序规则作了修改，它对各工件计算参量 $T_{i1}=\sum_{j=1}^{m}(m-j+1)t_{ij}$，并按递增顺序进行排列。首先选取队列中的第 1 个工件生成子调度，然后选取队列中的第 k 个工件尝试插入到已生成子调度的第 l 位置，其中 $[k/2]\leqslant l\leqslant k$，$k\geqslant 2$，进而令其中使得调度指标最小的方案为新的子调度。如此依次将剩余所有工件进行插入，从而得到一个次优调度。

这些启发式方法的优点是具有构造调度解的快速性，但调度质量还不是特别理想，其中属 NEH 方法和 Rajendran 方法的性能最好。基于此，优化质量较高的改进性搜索方法受到了重视，如遗传算法。

4.2 典型 Flow Shop 调度问题

鉴于 FSP 问题的重要性和代表性，许多研究工作者设计了若干典型问题(benchmarks)，用以测试和比较不同方法的优化性能，在此对其作简单介绍。

(1) Car 类。该类问题由 Carlier(1978)设计，包括 8 个不同规模的典型问题，分别命名为 Car1～Car8，表 4.2.1 给出了该类问题的部分信息。

(2) Hel 类。该类问题由 Heller(1960)给出，包括 2 个典型问题，分别命名为 Hel1 和 Hel2，表 4.2.2 给出了该类问题的部分信息。

表 4.2.1　Car 类问题

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Car1	11	5	7038	Carlier(1978)	7038	Carlier(1978)
Car2	13	4	7166	Carlier(1978)	7166	Carlier(1978)
Car3	12	5	7312	Carlier(1978)	7312	Carlier(1978)
Car4	14	4	8003	Carlier(1978)	8003	Carlier(1978)
Car5	10	6	7702	Carlier(1978)	7720	Carlier(1978)
Car6	8	9	8313	Carlier(1978)	8505	Carlier(1978)
Car7	7	7	6558	Carlier(1978)	6590	Carlier(1978)
Car8	8	8	8264	?	8366	?

表 4.2.2　Hel 类问题

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Hel1	100	10	上界 513 下界 510	Vaessens(1995)	513	Vaessens(1995)
Hel2	20	10	上界 134 下界 131	Vaessens(1996)	135	Vaessens(1995)

(3) Rec 类。该类问题由 Reeves(1995)给出,包括 7 个不同规模的 21 个典型问题,表 4.2.3 给出了该类问题的部分信息。

表 4.2.3　Rec 类问题的若干研究

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Rec01	20	5	1245	Vaessens(1995)	1247	Vaessens(1995)
Rec03	20	5	1093	Vaessens(1995)	1109	Reeves(1995)
Rec05	20	5	1228	Vaessens(1995)	1242	Vaessens(1995)
Rec07	20	10	上界 1545 下界 1514	Vaessens(1995, 结合 A&C 算法)	1566	Reeves(1995)
Rec09	20	10	上界 1530 下界 1519	Vaessens(1995, 结合 A&C 算法)	1537	Reeves(1995)
Rec11	20	10	1402	Vaessens(1995, 结合 A&C 算法)	1431	Reeves(1995)
Rec13	20	15	上界 1930 下界 1791	Vaessens(1995)	1930	Vaessens(1995)
Rec15	20	15	上界 1950 下界 1891	Vaessens(1995)	1950	Vaessens(1995)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Rec17	20	15	上界 1902 下界 1774	Reeves(1995)	1902	Reeves(1995)
Rec19	30	10	上界 2093 下界 2064	Vaessens(1995)	2093	Vaessens(1995)
Rec21	30	10	上界 2017 下界 2003	Vaessens(1995)	2017	Vaessens(1995)
Rec23	30	10	上界 2008 下界 1963	Vaessens(1996)	2011	Vaessens(1995)
Rec25	30	15	上界 2513 下界 2321	Vaessens(1995)	2513	Vaessens(1995)
Rec27	30	15	上界 2373 下界 2266	Vaessens(1995)	2373	Vaessens(1995)
Rec29	30	15	上界 2287 下界 2091	Vaessens(1995)	2287	Vaessens(1995)
Rec31	50	10	上界 3045 下界 3005	Vaessens(1995)	3045	Vaessens(1995)
Rec33	50	10	3093	Vaessens(1995，结合 A&C 算法)	3114	Vaessens(1995)
Rec35	50	10	3262	Vaessens(1995，结合 A&C 算法)	3277	Reeves(1995)
Rec37	75	20	上界 4951 下界 4764	Vaessens(1996，改进 N&S 的解)	上界 4951 下界 4890	Vaessens(1996，改进 N&S 的解)
Rec39	75	20	上界 5087 下界 4961	Reeves(1995)	上界 5087 下界 5043	Reeves(1995)
Rec41	75	20	上界 4960 下界 4721	Vaessens(1996，改进 N&S 的解)	上界 4960 下界 4910	Vaessens(1996，改进 N&S 的解)

注：Vaessens 采用分支定界方法，Reeves 采用遗传算法，Nowicki 和 Smutnicki 采用禁忌搜索；N&S 表示 Nowicki & Smutnicki，A&C 表示 Applegate & Cook 算法。

(4) TD 或 TA 类。该类问题由 Taillard 等(1993)给出，包括 12 个不同规模的 120 个典型问题，分别命名为 Ta001～Ta120，表 4.2.4 给出了该类问题的若干研究。

表 4.2.4 TD或TA类问题

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta001	20	5	1278	Taillard(1993)	1278	Taillard(1993)
Ta002	20	5	1358	Vaessens(1995，结合 A&C 算法)	1359	Taillard(1993)
Ta003	20	5	1073	Vaessens(1995，结合 A&C 算法)	1081	Taillard(1993)
Ta004	20	5	1292	Vaessens(1995，结合 A&C 算法)	1293	Taillard(1993)
Ta005	20	5	1231	Vaessens(1995，结合 A&C 算法)	1235	Taillard(1993)
Ta006	20	5	1193	Vaessens(1995，结合 A&C 算法)	1195	Taillard(1993)
Ta007	20	5	1234	Vaessens (1995)	1234	Taillard(1993)
Ta008	20	5	1199	Vaessens(1995，结合 A&C 算法)	1206	Taillard(1993)
Ta009	20	5	1210	Vaessens(1995，结合 A&C 算法)	1230	Taillard(1993)
Ta010	20	5	1103	Vaessens(1995，结合 A&C 算法)	1108	Taillard(1993)
Ta011	20	10	上界 1560 下界 1549	Vaessens(1995，结合 A&C 算法)	1582	Taillard(1993)
Ta012	20	10	上界 1644 下界 1603	Vaessens(1995，结合 A&C 算法)	1659	Taillard(1993)
Ta013	20	10	上界 1486 下界 1450	Vaessens(1995，结合 A&C 算法)	1496	Taillard(1993)
Ta014	20	10	上界 1368 下界 1356	Vaessens(1995，结合 A&C 算法)	1377	Taillard(1993)
Ta015	20	10	上界 1413 下界 1374	Vaessens(1995，结合 A&C 算法)	1491	Taillard(1993)
Ta016	20	10	上界 1369 下界 1347	Vaessens(1995，结合 A&C 算法)	1397	Taillard(1993)
Ta017	20	10	上界 1428 下界 1400	Vaessens(1995，结合 A&C 算法)	1484	Taillard(1993)
Ta018	20	10	上界 1527 下界 1446	Vaessens(1995，结合 A&C 算法)	1538	Taillard(1993)
Ta019	20	10	上界 1586 下界 1558	Vaessens(1995，结合 A&C 算法)	1539	Taillard(1993)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta020	20	10	上界 1559 下界 1525	Vaessens(1995，结合 A&C 算法)	1591	Taillard(1993)
Ta021	20	20	上界 2293 下界 2021	Vaessens(1995，结合 A&C 算法)	2297	Taillard(1993)
Ta022	20	20	上界 2092 下界 1847	Vaessens(1995，结合 A&C 算法)	2099	Taillard(1993)
Ta023	20	20	上界 2313 下界 2006	Vaessens(1995，结合 A&C 算法)	2326	Taillard(1993)
Ta024	20	20	上界 2223 下界 2001	Taillard(1993)	2223	Taillard(1993)
Ta025	20	20	上界 2291 下界 2086	Taillard(1993)	2291	Taillard(1993)
Ta026	20	20	上界 2221 下界 1971	Vaessens(1995，结合 A&C 算法)	2226	Taillard(1993)
Ta027	20	20	上界 2267 下界 2010	Vaessens(1995，结合 A&C 算法)	2273	Taillard(1993)
Ta028	20	20	上界 2183 下界 1961	Vaessens(1995，结合 A&C 算法)	2200	Taillard(1993)
Ta029	20	20	上界 2227 下界 1941	Vaessens(1995，结合 A&C 算法)	2237	Taillard(1993)
Ta030	20	20	上界 2178 下界 1992	Taillard(1993)	2178	Taillard(1993)
Ta031	50	5	2724	Taillard(1993)	2724	Taillard(1993)
Ta032	50	5	2834	Nowicki, Smutnicki (1996)	2834	Nowicki, Smutnicki (1996)
Ta033	50	5	2612	Vaessens (1995)	2621	Taillard(1993)
Ta034	50	5	2751	Taillard(1993)	2751	Taillard(1993)
Ta035	50	5	2853	Vaessens(1995，结合 A&C 算法)	2863	Taillard(1993)
Ta036	50	5	2825	Vaessens(1995，结合 A&C 算法)	2829	Taillard(1993)
Ta037	50	5	上界 2716 下界 2715	Vaessens (1995)	2725	Taillard(1993)
Ta038	50	5	2683	Taillard(1993)	2683	Taillard(1993)
Ta039	50	5	上界 2545 下界 2542	Vaessens (1995)	2552	Nowicki, Smutnicki (1996)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta040	50	5	2776	Vaessens(1995，结合 A&C 算法)	2782	Taillard(1993)
Ta041	50	10	上界 2991 下界 2970	Vaessens (1995)	2991	Vaessens (1995)
Ta042	50	10	上界 2867 下界 2829	Vaessens (1995)	2867	Vaessens (1995)
Ta043	50	10	上界 2832 下界 2828	Vaessens(1995，结合 A&C 算法)	2839	Vaessens (1995)
Ta044	50	10	上界 3063 下界 3039	Vaessens (1995)	3063	Vaessens (1995)
Ta045	50	10	上界 2976 下界 2935	Vaessens (1995)	2976	Vaessens (1995)
Ta046	50	10	上界 2991 下界 2981	Vaessens(1995，结合 A&C 算法)	3006	Nowicki，Smutnicki (1996)
Ta047	50	10	3093	Vaessens (1995)	3093	Vaessens (1995)
Ta048	50	10	上界 3026 下界 3001	Vaessens (1995)	3037	Vaessens (1995)
Ta049	50	10	上界 2887 下界 2858	Vaessens (1995)	2897	Vaessens (1995)
Ta050	50	10	上界 3065 下界 3046	Vaessens (1995)	3065	Vaessens (1995)
Ta051	50	20	上界 3856 下界 3566	Nowicki，Smutnicki (1996)	上界 3856 下界 3771	Nowicki，Smutnicki (1996)
Ta052	50	20	上界 3707 下界 3541	Vaessens(1996，改进 N&S 的解)	上界 3707 下界 3668	Vaessens(1996，改进 N&S 的解)
Ta053	50	20	上界 3643 下界 3421	Vaessens(1996，改进 N&S 的解)	上界 3643 下界 3591	Vaessens(1996，改进 N&S 的解)
Ta054	50	20	上界 3731 下界 3401	Vaessens (1995)	上界 3731 下界 3635	Vaessens (1995)
Ta055	50	20	上界 3619 下界 3378	Nowicki，Smutnicki (1996)	上界 3619 下界 3553	Nowicki，Smutnicki (1996)
Ta056	50	20	上界 3687 下界 3511	Vaessens (1995)	上界 3687 下界 3667	Vaessens (1995)
Ta057	50	20	上界 3706 下界 3474	Vaessens(1996，改进 N&S 的解)	上界 3706 下界 3672	Vaessens(1996，改进 N&S 的解)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta058	50	20	上界 3700 下界 3454	Nowicki, Smutnicki (1996)	上界 3700 下界 3627	Nowicki, Smutnicki (1996)
Ta059	50	20	上界 3755 下界 3500	Vaessens(1996, 改进 N&S 的解)	上界 3755 下界 3645	Vaessens(1996, 改进 N&S 的解)
Ta060	50	20	上界 3767 下界 3521	Nowicki, Smutnicki (1996)	上界 3767 下界 3696	Nowicki, Smutnicki (1996)
Ta061	100	5	5493	Taillard(1993)	5493	Taillard(1993)
Ta062	100	5	5257	Vaessens (1995)	5268	Nowicki, Smutnicki (1996)
Ta063	100	5	上界 5173 下界 5171	Vaessens (1995)	5175	Taillard(1993)
Ta064	100	5	4993	Vaessens (1995)	5014	Nowicki, Smutnicki (1996)
Ta065	100	5	上界 5247 下界 5244	Vaessens(1995, 结合 A&C 算法)	5250	Taillard(1993)
Ta066	100	5	5135	Taillard(1993)	5135	Taillard(1993)
Ta067	100	5	5232	Vaessens(1995, 结合 A&C 算法)	5246	Nowicki, Smutnicki (1996)
Ta068	100	5	5083	Vaessens (1995)	5094	Taillard(1993)
Ta069	100	5	5442	Vaessens(1995, 结合 A&C 算法)	5448	Taillard(1993)
Ta070	100	5	5318	Vaessens(1995, 结合 A&C 算法)	5322	Vaessens (1995)
Ta071	100	10	5759	Vaessens (1995)	5770	Nowicki, Smutnicki (1996)
Ta072	100	10	上界 5349 下界 5345	Nowicki, Smutnicki (1996)	5349	Nowicki, Smutnicki (1996)
Ta073	100	10	上界 5673 下界 5654	Vaessens(1995, 结合 A&C 算法)	5676	Vaessens (1995)
Ta074	100	10	上界 5759 下界 5741	Vaessens(1995, 结合 A&C 算法)	5781	Vaessens (1995)
Ta075	100	10	上界 5455 下界 5432	Vaessens(1995, 结合 A&C 算法)	5467	Vaessens (1995)
Ta076	100	10	5293	Vaessens(1995, 结合 A&C 算法)	5303	Nowicki, Smutnicki (1996)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta077	100	10	上界 5584 下界 5557	Vaessens(1995, 结合 A&C 算法)	5595	Vaessens (1995)
Ta078	100	10	上界 5617 下界 5604	Vaessens (1995)	5617	Vaessens (1995)
Ta079	100	10	上界 5852 下界 5843	Vaessens(1995, 结合 A&C 算法)	5871	Vaessens (1995)
Ta080	100	10	5845	Nowicki, Smutnicki (1996)	5845	Nowicki, Smutnicki (1996)
Ta081	100	20	上界 6228 下界 5926	Vaessens(1996, 改进 N&S 的解)	上界 6228 下界 6106	Vaessens(1996, 改进 N&S 的解)
Ta082	100	20	上界 6210 下界 6122	Nowicki, Smutnicki (1996)	上界 6210 下界 6183	Nowicki, Smutnicki (1996)
Ta083	100	20	上界 6271 下界 6166	Vaessens(1996, 改进 N&S 的解)	上界 6271 下界 6252	Vaessens(1996, 改进 N&S 的解)
Ta084	100	20	上界 6269 下界 6136	Vaessens(1996, 改进 N&S 的解)	上界 6269 下界 6254	Vaessens(1996, 改进 N&S 的解)
Ta085	100	20	上界 6319 下界 6166	Vaessens(1996, 改进 N&S 的解)	上界 6319 下界 6262	Vaessens(1996, 改进 N&S 的解)
Ta086	100	20	上界 6403 下界 6203	Vaessens(1996, 改进 N&S 的解)	上界 6403 下界 6302	Vaessens(1996, 改进 N&S 的解)
Ta087	100	20	上界 6292 下界 6051	Vaessens(1996, 改进 N&S 的解)	上界 6292 下界 6184	Vaessens(1996, 改进 N&S 的解)
Ta088	100	20	上界 6423 下界 6146	Nowicki, Smutnicki (1996)	上界 6423 下界 6315	Nowicki, Smutnicki (1996)
Ta089	100	20	上界 6275 下界 6041	Vaessens(1996, 改进 N&S 的解)	上界 6275 下界 6204	Vaessens(1996, 改进 N&S 的解)
Ta090	100	20	上界 6434 下界 6381	Vaessens(1996, 改进 N&S 的解)	上界 6434 下界 6404	Vaessens(1996, 改进 N&S 的解)
Ta091	200	10	10857	Vaessens(1995, 结合 A&C 算法)	10862	Vaessens (1995)
Ta092	200	10	上界 10480 下界 10430	Vaessens (1995)	10480	Vaessens (1995)
Ta093	200	10	上界 10922 下界 10914	Nowicki, Smutnicki (1996)	10922	Nowicki, Smutnicki (1996)
Ta094	200	10	上界 10862 下界 10859	Vaessens(1995, 结合 A&C 算法)	10889	Nowicki, Smutnicki (1996)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta095	200	10	上界 10524 下界 10484	Nowicki, Smutnicki (1996)	10524	Nowicki, Smutnicki (1996)
Ta096	200	10	10329	Vaessens (1995)	10329	Vaessens (1995)
Ta097	200	10	10836	Vaessens(1995, 结合 A&C 算法)	10854	Vaessens (1995)
Ta098	200	10	上界 10727 下界 10709	Vaessens(1995, 结合 A&C 算法)	10730	Vaessens (1995)
Ta099	200	10	10419	Vaessens (1995)	10438	Nowicki, Smutnicki (1996)
Ta100	200	10	10675	Vaessens (1995)	10675	Vaessens (1995)
Ta101	200	20	上界 11195 下界 10979	Vaessens(1996, 改进 N&S 的解)	上界 11195 下界 11152	Vaessens(1996, 改进 N&S 的解)
Ta102	200	20	上界 11223 下界 10947	Vaessens(1996, 改进 N&S 的解)	上界 11223 下界 11143	Vaessens(1996, 改进 N&S 的解)
Ta103	200	20	上界 11337 下界 11150	Vaessens(1996, 改进 N&S 的解)	上界 11337 下界 11281	Vaessens(1996, 改进 N&S 的解)
Ta104	200	20	上界 11299 下界 11127	Vaessens(1996, 改进 N&S 的解)	上界 11299 下界 11275	Vaessens(1996, 改进 N&S 的解)
Ta105	200	20	上界 11260 下界 11132	Vaessens(1996, 改进 N&S 的解)	上界 11260 下界 11259	Vaessens(1996, 改进 N&S 的解)
Ta106	200	20	上界 11189 下界 11085	Vaessens(1996, 改进 N&S 的解)	上界 11189 下界 11176	Vaessens(1996, 改进 N&S 的解)
Ta107	200	20	上界 11386 下界 11194	Vaessens(1996, 改进 N&S 的解)	上界 11386 下界 11337	Vaessens(1996, 改进 N&S 的解)
Ta108	200	20	上界 11334 下界 11126	Vaessens(1996, 改进 N&S 的解)	上界 11334 下界 11301	Vaessens(1996, 改进 N&S 的解)
Ta109	200	20	上界 11192 下界 10965	Vaessens(1996, 改进 N&S 的解)	上界 11192 下界 11145	Vaessens(1996, 改进 N&S 的解)
Ta110	200	20	上界 11313 下界 11122	Nowicki, Smutnicki (1996)	上界 11313 下界 11284	Nowicki, Smutnicki (1996)
Ta111	500	20	上界 26059 下界 25922	Vaessens(1996, 改进 N&S 的解)	上界 26059 下界 26040	Vaessens(1996, 改进 N&S 的解)
Ta112	500	20	上界 26520 下界 26353	Vaessens (1995)	上界 26520 下界 26500	Vaessens (1995)
Ta113	500	20	上界 26371 下界 26320	Vaessens (1995)	26371	Vaessens (1995)

续表

问题	n	m	Flow Shop 最优 Makespan	最早取得最优值的研究者	置换 Flow Shop 最优 Makespan	最早取得最优值的研究者
Ta114	500	20	上界 26456 下界 26424	Vaessens (1995)	26456	Vaessens (1995)
Ta115	500	20	上界 26334 下界 26181	Vaessens (1995)	26334	Vaessens (1995)
Ta116	500	20	上界 26477 下界 26401	Vaessens (1995)	26477	Vaessens (1995)
Ta117	500	20	上界 26389 下界 26300	Vaessens (1995)	26389	Vaessens (1995)
Ta118	500	20	上界 26560 下界 26429	Vaessens (1995)	26560	Vaessens (1995)
Ta119	500	20	上界 26005 下界 25891	Vaessens (1995)	26005	Vaessens (1995)
Ta120	500	20	上界 26457 下界 26315	Vaessens (1995)	26457	Vaessens (1995)

上述各类典型 FSP 问题，可由网址 http://mscmga.ms.ic.ac.uk 下载，本书附录 2 列出各调度问题的具体加工数据。

Flow Shop 调度问题是 Job Shop 调度的一个特例，因此许多解决 Job Shop 调度的遗传操作和遗传算法框架都可以直接或间接用于求解 Flow Shop 问题。同时，鉴于本质上置换 Flow Shop 问题等同于旅行商问题，因此许多求解 TSP 问题的遗传操作都可直接或间接用于求解置换 Flow Shop 问题。譬如，PMX，OX，LOX，NABEL 等交叉操作，SWAP，INV，INS 等变异操作。因此，本章不再重复介绍遗传算法的操作和框架设计，读者可参阅本书第 2 章和第 3 章相关内容。

4.3 置换 Flow Shop 调度的遗传算法

本节介绍置换 Flow Shop 调度问题 $n/m/P/C_{max}$ 的一种改进遗传算法设计。首先探讨若干遗传操作对传统 GA(SGA)性能的影响，其中 SGA 采用如下设计方案：

- 置换编码，即所有工件的排列；
- 随机初始种群，或结合 NEH 和随机初始化(记 SGA+NEH)；
- 交叉操作可以采用 LOX，PMX，C1，NABEL 等操作，交叉操作对当前种群中的最优个体和另一个随机选取的个体来进行，并重复 $P_s/2$ 次(P_s 为种群规模)，然后保留新旧个体中最好的 P_s 个个体进行后续的变异操作；

• 变异操作可以采用 SWAP,INV 和 INS 等操作,整个搜索进程保留 best so far 解;
• 终止准则采用固定进化代数,考虑到问题规模的影响,取最大进化代数为 $n\times m$。

同时,利用 29 个 Benchmark 问题进行仿真研究,包括 8 个 Car 类问题和 21 个 Rec 类问题,具体加工数据见附录 2。

4.3.1 初始化对 SGA 的影响

取种群规模为 40,交叉概率为 1,变异概率为 0.05,使用 LOX 交叉和 SWAP 变异来探讨初始化对算法性能的影响,20 次随机仿真实验的统计数据见表 4.3.1。

表 4.3.1 初始化对 SGA 的影响

问题	n	m	C^*	NEH	SGA			SGA+NEH		
				RE	BRE	ARE	ART	BRE	ARE	ART
Car1	11	5	7038	0	0	0.27	0.03	0	0	0.03
Car2	13	4	7166	2.931	0	4.07	0.02	2.93	2.93	0.03
Car3	12	5	7312	1.792	1.19	2.95	0.03	0.82	1.21	0.03
Car4	14	4	8003	0.387	0	2.36	0.03	0	0.07	0.03
Car5	10	6	7720	4.236	0	1.46	0.04	0	1.14	0.03
Car6	8	9	8505	3.621	0	1.86	0.05	0	2.82	0.04
Car7	7	7	6590	6.343	0	1.57	0.03	0	1.36	0.03
Car8	8	8	8366	1.088	0	2.59	0.04	0	0.03	0.04
Rec01	20	5	1247	8.42	2.81	6.96	0.08	2.25	6.13	0.08
Rec03	20	5	1109	6.583	1.89	4.45	0.08	1.26	4.27	0.08
Rec05	20	5	1242	4.831	1.93	3.82	0.09	2.33	2.90	0.09
Rec07	20	10	1566	5.364	1.15	5.31	0.29	3.38	5.27	0.29
Rec09	20	10	1537	6.766	3.12	4.73	0.28	0.39	2.13	0.29
Rec11	20	10	1431	8.246	3.91	7.39	0.28	1.19	3.66	0.30
Rec13	20	15	1930	7.617	3.68	5.97	0.60	1.92	4.41	0.62
Rec15	20	15	1950	4.923	2.21	4.29	0.60	2.87	4.02	0.61
Rec17	20	15	1902	7.466	3.15	6.08	0.58	2.16	4.02	0.63
Rec19	30	10	2093	6.541	4.01	6.07	0.61	2.05	4.35	0.62
Rec21	30	10	2017	4.561	3.42	6.07	0.60	3.52	3.58	0.62
Rec23	30	10	2011	9.995	3.83	7.46	0.61	3.63	5.12	0.63
Rec25	30	15	2513	6.964	4.42	7.20	1.29	3.14	4.89	1.35
Rec27	30	15	2373	8.512	4.93	6.85	1.30	3.16	5.21	1.35
Rec29	30	15	2287	5.422	6.21	8.48	1.29	3.32	4.93	1.36

续表

问题	n	m	C^*	NEH	SGA			SGA+NEH		
				RE	BRE	ARE	ART	BRE	ARE	ART
ec31	50	10	3045	10.279	6.17	8.02	1.66	5.94	6.66	1.73
Rec33	50	10	3114	4.753	3.08	5.12	1.62	2.70	3.38	1.68
Rec35	50	10	3277	5.005	1.46	3.30	1.65	1.89	2.58	1.74
Rec37	75	20	4890	9.141	7.89	10.07	13.52	7.14	7.94	14.79
Rec39	75	20	5043	8.646	7.32	8.51	13.64	6.25	7.09	14.82
Rec41	75	20	4910	10.692	8.51	10.03	13.65	7.49	8.47	14.76

由表4.3.1可见，纯随机搜索的SGA很容易早熟收敛，甚至有时其性能劣于NEH启发式方法；结合NEH初始化的SGA尽管偶尔其性能会劣于纯随机初始化的结果，只可以理解为NEH为SGA提供了一个较深的波谷，即局部较小，但这种可能性较小，同时其性能一致性地优于NEH方法，而且搜索时间仅有细微增加。因此，在后面介绍的改进遗传算法中采用这种混合初始化。

4.3.2 交叉操作对SGA的影响

采用上面相同的算法参数(采用SWAP变异)探讨4种交叉操作对SGA性能的影响，20次随机仿真实验的统计数据见表4.3.2。

表4.3.2 交叉操作对SGA的影响

问题	LOX			PMX			C1			NABEL		
	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART
Car1	0	0.27	0.03	0	0.51	0.03	0	0.27	0.03	0	2.60	0.03
Car2	0	4.07	0.02	0	3.91	0.03	0	3.92	0.03	2.93	7.34	0.03
Car3	1.19	2.95	0.03	1.79	3.54	0.03	1.20	3.25	0.03	3.90	6.60	0.03
Car4	0	2.36	0.03	0	3.56	0.04	0	2.81	0.04	2.56	5.34	0.02
Car5	0	1.46	0.04	0.38	1.50	0.03	0.38	1.66	0.04	0.38	2.47	0.03
Car6	0	1.86	0.05	0	2.53	0.05	0	2.16	0.05	0.76	4.15	0.04
Car7	0	1.57	0.03	0	2.19	0.02	0	1.18	0.02	0	1.46	0.02
Car8	0	2.59	0.04	0	2.99	0.04	0	1.70	0.04	1.09	2.41	0.03
Rec01	2.81	6.96	0.08	2.89	6.77	0.08	4.49	6.38	0.16	6.58	10.14	0.08
Rec03	1.89	4.45	0.08	2.16	5.13	0.08	0.18	3.55	0.16	5.41	8.28	0.08
Rec05	1.93	3.82	0.09	0.24	3.15	0.08	2.17	3.38	0.16	2.66	5.82	0.08
Rec07	1.15	5.31	0.29	1.15	5.92	0.27	1.72	4.92	0.31	5.94	8.02	0.26
Rec09	3.12	4.73	0.28	3.38	6.01	0.27	2.67	4.86	0.31	8.78	10.17	0.26
Rec11	3.91	7.39	0.28	2.66	7.02	0.28	2.73	6.12	0.32	9.08	12.45	0.26

续表

问题	LOX			PMX			C1			NABEL		
	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART
Rec13	3.68	5.97	0.60	2.95	5.24	0.58	2.80	5.30	0.63	8.86	10.55	0.57
Rec15	2.21	4.29	0.60	1.64	4.13	0.58	1.85	4.52	0.64	6.72	8.89	0.56
Rec17	3.15	6.08	0.58	3.31	6.46	0.58	2.68	5.75	0.63	9.04	12.0	0.56
Rec19	4.01	6.07	0.61	4.83	7.13	0.58	3.87	6.13	0.66	10.89	12.93	0.63
Rec21	3.42	6.07	0.60	4.36	7.48	0.58	2.43	5.50	0.65	10.31	13.06	0.59
Rec23	3.83	7.46	0.61	3.83	7.15	0.59	3.08	5.97	0.66	11.69	13.36	0.55
Rec25	4.42	7.20	1.29	4.42	4.64	1.26	5.21	6.73	1.39	9.87	12.35	1.20
Rec27	4.93	6.85	1.30	3.71	6.88	1.26	4.59	6.62	1.41	12.22	14.32	1.21
Rec29	6.21	8.48	1.29	4.81	9.43	1.26	5.77	8.67	1.41	12.11	16.32	1.21
Rec31	6.17	8.02	1.66	6.67	8.15	1.63	6.01	7.88	1.94	13.63	14.45	1.43
Rec33	3.08	5.12	1.62	3.92	5.69	1.64	2.34	4.77	1.93	10.02	11.91	1.41
Rec35	1.46	3.30	1.65	1.10	3.75	1.64	2.44	3.42	1.95	6.50	8.24	1.43
Rec37	7.89	10.07	13.52	8.88	10.66	13.19	8.04	10.06	15.49	16.85	18.38	12.29
Rec39	7.32	8.51	13.64	6.88	8.51	13.17	7.28	8.56	15.62	15.41	16.32	12.37
Rec41	8.51	10.03	13.65	8.62	10.08	13.19	8.13	10.18	15.65	17.94	19.17	12.41

由表 4.3.2 可见，NABEL 操作下的 SGA 虽然搜索时间最短，但搜索质量最差，其原因是父代个体被严重打乱，有效模式没有被有效继承；另外 3 种操作中，C1 操作花费的时间最多，但三者在统计意义下的搜索质量没有较大的区别。对于一个未知问题，通常是事先不知道哪种操作最好，因此在后面的改进遗传算法中采用这些操作的混合应用。

4.3.3 变异操作对 SGA 的影响

采用上面相同的算法参数（采用 LOX 交叉）探讨 3 种变异操作对 SGA 性能的影响，20 次随机仿真实验的统计数据见表 4.3.3。

表 4.3.3 变异操作对 SGA 的影响

问题	SWAP			INV			INS		
	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART
Car1	0	0.27	0.03	0	0.42	0.03	0	0.06	0.03
Car2	0	4.07	0.02	0.29	4.52	0.03	0	3.48	0.03
Car3	1.19	2.95	0.03	1.19	3.03	0.04	1.19	2.90	0.04
Car4	0	2.36	0.03	0	1.91	0.03	0	1.51	0.03
Car5	0	1.46	0.04	0.61	2.08	0.04	0	2.17	0.03

续表

问题	SWAP			INV			INS		
	BRE	ARE	ART	BRE	ARE	ART	BRE	ARE	ART
Car6	0	1.86	0.05	0	2.47	0.06	0	2.43	0.03
Car7	0	1.57	0.03	0	1.57	0.02	0	1.30	0.03
Car8	0	2.59	0.04	0	3.31	0.04	0.65	2.68	0.01
Rec01	2.81	6.96	0.08	3.37	7.83	0.09	2.00	6.68	0.09
Rec03	1.89	4.45	0.08	0.99	6.06	0.08	2.25	5.08	0.08
Rec05	1.93	3.82	0.09	1.93	3.90	0.09	1.13	3.09	0.08
Rec07	1.15	5.31	0.29	3.38	7.13	0.29	3.38	6.16	0.29
Rec09	3.12	4.73	0.28	2.54	7.07	0.29	2.80	6.24	0.28
Rec11	3.91	7.39	0.28	5.03	9.39	0.29	2.66	6.70	0.28
Rec13	3.68	5.97	0.60	4.97	7.40	0.61	3.16	6.02	0.61
Rec15	2.21	4.29	0.60	3.59	6.10	0.61	2.31	4.81	0.60
Rec17	3.15	6.08	0.58	4.89	8.64	0.60	3.47	7.19	0.60
Rec19	4.01	6.07	0.61	5.16	8.17	0.61	4.49	7.04	0.61
Rec21	3.42	6.07	0.60	3.62	8.65	0.62	2.88	5.96	0.60
Rec23	3.83	7.46	0.61	5.32	8.73	0.62	4.43	7.52	0.62
Rec25	4.42	7.20	1.29	6.29	8.40	1.31	4.10	6.04	1.30
Rec27	4.93	6.85	1.30	5.82	9.72	1.32	4.72	7.22	1.31
Rec29	6.21	8.48	1.29	8.22	12.03	1.31	6.08	9.81	1.31
Rec31	6.17	8.02	1.66	6.86	10.56	1.67	6.34	8.10	1.67
Rec33	3.08	5.12	1.62	4.75	7.08	1.65	3.21	5.11	1.64
Rec35	1.46	3.30	1.65	3.54	5.02	1.66	2.17	3.40	1.67
Rec37	7.89	10.07	13.52	11.68	13.38	13.79	8.73	10.97	13.69
Rec39	7.32	8.51	13.64	10.01	12.02	13.86	6.54	8.52	15.41
Rec41	8.51	10.03	13.65	12.40	14.41	13.87	7.94	10.28	15.55

由表4.3.3可见，3种变异操作在统计意义下的搜索质量没有较大的区别，因此在后面的改进遗传算法中也采用这些操作的混合应用。

4.3.4 改进遗传算法

基于上述研究，为了改善SGA算法的性能，在此提出一种改进遗传算法。

首先，理论上GA的全局收敛性保证了算法对初值的鲁棒性，但在实际应用时由于收敛条件难以保证，从而导致算法的优化性能和效率对初始种群的依赖性。因此，采用启发式方法（如NEH法）和随机方法共同产生初始种群，保证了初始种群一定的质量，并不失初始种群的多样性，同时启发式方法的快速性保证了这种初始化的速度。

其次，GA 的优化过程中交叉操作通常是作用在整个种群上，且结构一成不变。在此将种群分解为若干子种群（类似于小生群概念），各子种群分别用不同机制的交叉操作来实现进化，用复合多交叉方式来继承父代的优良模式，并一定程度上保持种群的多样性，从而在多种邻域结构下丰富了交叉环节的搜索行为和机制，同时增强全空间和局部范围的搜索能力。

再者，采用种群整体替换策略，将作用在不同子种群上的交叉操作产生的所有新个体与父代种群进行整体择优筛选，从而加速种群的进化过程。

尤其是，GA 的复制对当前种群外的解空间无探索能力，个体分布"畸形"时交叉的进化能力有限，小概率变异很难增加种群的多样性，若收敛准则设计不好，GA 经常会出现进化缓慢或"早熟"收敛现象。因此，将 SA 基于概率突跳的 Metropolis 抽样过程融入 GA，即将可控性概率劣向转移的避免陷入局部极小的机制融入 GA，并且抽样方式又能够体现为另一种邻域搜索。同时，Metropolis 抽样过程起到概率可控的变异操作，控制初温可控制初始搜索行为，控制温度的高低可控制突跳能力的强弱，高温下的强突跳性有利于避免陷入局部极小，低温下的趋化性有利于提高局部搜索能力，控制降温速率可控制突跳能力的下降幅度，控制抽样次数可控制各温度下搜索能力。这样，不仅丰富了 GA 的搜索行为，而且避免了变异概率难以选取的困难。

由此，构造如图 4.3.1 所示的一种改进遗传算法。

值得一提的是，该改进算法很容易采用并行机或分布式方式处理，并且可保持遗传算法的通用性，可以在多种领域加以应用。对于置换 Flow Shop 问题，该算法的参数和操作具体设计如下。

(1) 初始种群。采用 NEH 法产生一个初始解，同时随机地产生其他个体，来共同组成初始种群，仿真时种群数取 10。

(2) 初温。当初始种群产生后，算法确定其中的最优和最差状态（目标值分别为 c_b 和 c_w），并令最差状态相对最优状态的接受概率为 p_r，由 $t_0=-(c_w-c_b)/\ln p_r$ 可确定初温，仿真时取 $p_r=0.1$。

(3) 交叉操作。在执行交叉操作前，将整个种群分成 K 个子种群（仿真时均分为 4 份），在各子种群中按一定的概率随机选取一个个体与整个种群中的最佳个体以不同的方式进行交叉，直至产生 K 个新子种群，然后进行替换操作。仿真时采用 LOX，C1，PMX，NABEL 这 4 种不同方式的交叉操作来继承父代优良模式，其中 LOX 能够尽量保留基因间的相对位置和相对染色体顶端的绝对位置，C1 能够在不过分打乱染色体的基础上提供足够的修改范围，PMX 能够一定程度上满足模式定理使最佳模式得以最大可能保留，NABEL 采用群的置换操作来快速产生新个体并使旧个体发生很大的修改，如此复合化多交叉操作使得搜索行为具有明显的多样性。

(4) 种群替换。采用种群整体替换策略，将作用在不同子种群上的交叉操作产生的所有新个体与父代种群进行整体择优筛选，从而加速种群的进化过程。

(5) Metropolis 抽样过程。抽样过程是针对每个个体进行的，对旧个体采用互

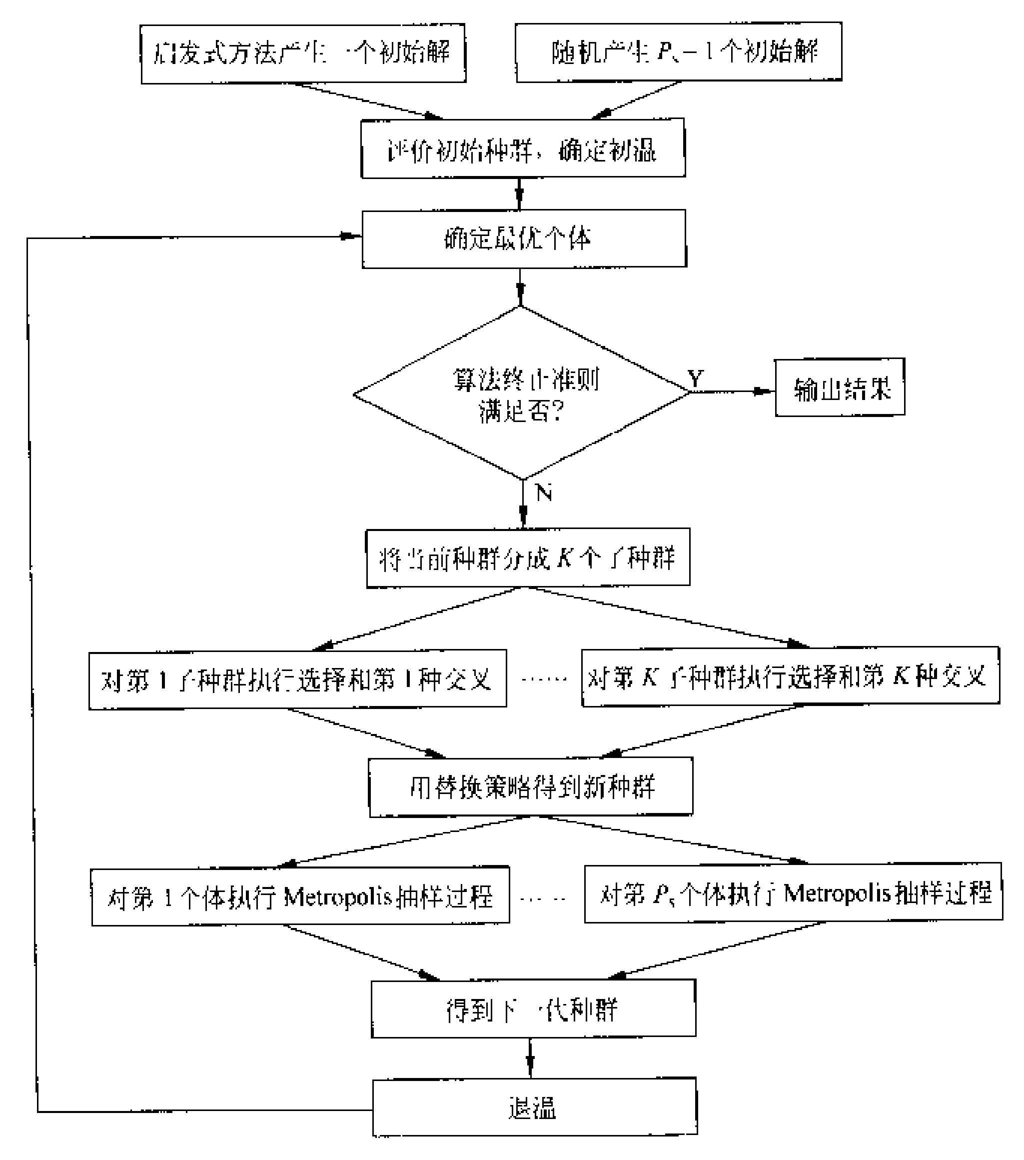


图 4.3.1　Flow Shop 调度的改进遗传算法的流程图

换 SWAP 方式产生新个体，并通过判断 $\min\{1,\exp(-\Delta/t)\}>random[0,1]$ 来接受新状态(式中 t 为温度，Δ 为新旧个体的目标值差)，如此起到概率可控的变异操作，而且可以做到增加种群多样性并避免搜索陷入局部极小。考虑到搜索空间大小 $n!$ 受工件数 n 的影响，仿真时抽样过程执行 n 步，并及时更新“best so far”的个体，以免遗失最优解。

(6) 退温。指数退温，即 $t_k=\lambda t_{k-1}$，仿真时取 $\lambda=0.95$。

(7) 终止准则。同前，如此可在相同计算量与 SGA 进行比较。

4.3.5　数值仿真与分析

改进遗传算法和 SGA 对各算例均随机运行 20 次(GA 采用 LOX 交叉，变异概率 0.05，初始种群随机产生)，仿真统计结果见表 4.3.4。表中，C^* 为问题的最优解；

RE 表示与 C^* 的相对误差$(C_{max}^{Heu}-C^*)/C^*\times 100\%$;BRE 为最优相对误差;ARE 为平均相对误差;WRE 为最差相对误差。图 4.3.2 显示了改进 GA,SGA 的结果相对 NEH 结果的偏差,即$(C_{max}^{GA}-C_{max}^{NEH})/C_{max}^{NEH}\times 100\%$,横轴为算例序号。

表 4.3.4 仿真统计结果比较

问题	n	m	C^*	改进 GA			NEH	SGA	
				BRE	ARE	WRE	RE	BRE	ARE
Car1	11	5	7038	0	0	0	0	0	0.27
Car2	13	4	7166	0	0	0	2.93	0	4.07
Car3	12	5	7312	0	0	0	1.79	1.19	2.95
Car4	14	4	8003	0	0	0	0.39	0	2.36
Car5	10	6	7720	0	0.08	0.61	4.24	0	1.46
Car6	8	9	8505	0	0.24	0.76	3.62	0	1.86
Car7	7	7	6590	0	0	0	6.34	0	1.57
Car8	8	8	8366	0	0	0	1.09	0	2.59
Rec01	20	5	1247	0	0.14	0.16	8.42	2.81	6.96
Rec03	20	5	1109	0	0.14	0.18	6.58	1.89	4.45
Rec05	20	5	1242	0	0.31	1.53	4.83	1.93	3.82
Rec07	20	10	1566	0	0.59	1.15	5.36	1.15	5.31
Rec09	20	10	1537	0	0.79	2.41	6.77	3.12	4.73
Rec11	20	10	1431	0	1.48	2.37	8.25	3.91	7.39
Rec13	20	15	1930	0.62	1.52	3.16	7.62	3.68	5.97
Rec15	20	15	1950	0.46	1.28	2.87	4.92	2.21	4.29
Rec17	20	15	1902	1.73	2.69	3.68	7.47	3.15	6.08
Rec19	30	10	2093	1.09	1.58	2.39	6.64	4.01	6.07
Rec21	30	10	2017	1.44	1.52	1.64	4.56	3.42	6.07
Rec23	30	10	2011	0.45	0.99	1.74	10.0	3.83	7.46
Rec25	30	15	2513	1.63	2.74	3.94	6.96	4.42	7.20
Rec27	30	15	2373	0.80	2.11	3.33	8.51	4.93	6.85
Rec29	30	15	2287	1.53	2.59	3.72	5.42	6.21	8.48
Rec31	50	10	3045	0.49	1.62	2.82	10.28	6.17	8.02
Rec33	50	10	3114	0.13	0.75	0.83	4.75	3.08	5.12
Rec35	50	10	3277	0	0	0	5.01	1.46	3.30
Rec37	75	20	4951	2.26	3.49	4.34	7.80	6.56	8.72
Rec39	75	20	5087	1.14	1.93	3.58	7.71	6.39	7.57
Rec41	75	20	4960	3.27	3.78	4.69	9.58	7.42	8.92

由仿真结果可见:

(1) 改进 GA 具有很好的优化质量,对 20×10(尤其以下)规模的问题均能够得

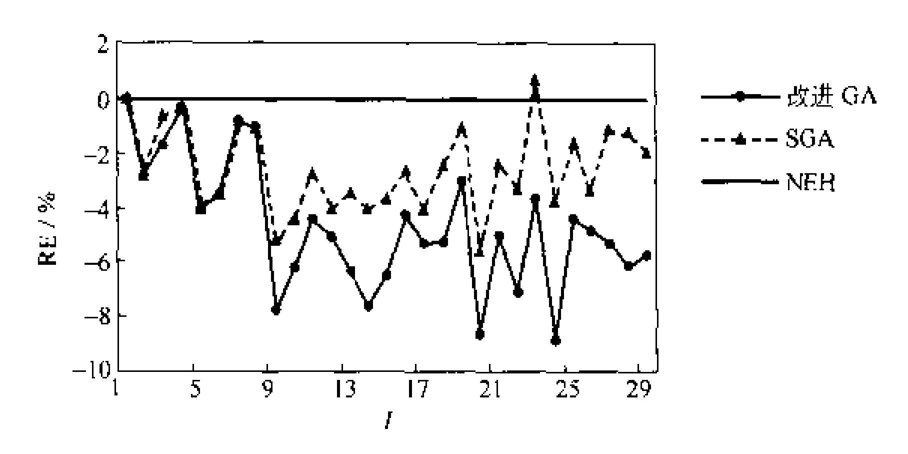


图 4.3.2 改进 GA 和 SGA 与 NEH 方法的相对性能

到最优解,甚至是 100%的最优解;对较大规模的问题能够得到良好的近似最优解,且质量大大优于传统 SGA 和 NEH 方法。

(2) 改进 GA 的 ARE 指标很小,这反映了算法较好的初值鲁棒性。

(3) 改进 GA 的自身波动性很小,其最差的优化质量与最佳优化质量差距不太大,而且最差性能也全面地大大优于 NEH 方法,并对较大规模问题还大幅度地优于传统 SGA,而传统 SGA 对 NEH 的性能改进幅度较小,甚至会产生差于 NEH 的解。

此外,韦有双等(2000)的改进启发式方法、Osman 等(1989)的模拟退火、Widmer 等(1989)的禁忌搜索的优化结果均还不能一致地优于 NEH 方法,但改进 GA 能够 100%地大幅度优于 NEH 方法,甚至改进 GA 的最差结果也远远优于 NEH 方法,由此可见改进 GA 在优化质量方面相对 NEH 和传统 SGA,SA 的优越性。

总之,改进 GA 之所以表现出良好优化能力,其原因是:结合启发式方法和随机方法有利于产生较好质量并保持一定多样性的初始种群;采用复合化多交叉操作不仅能够很好地继承父代优良模式,进行全空间的有效搜索,而且也能增加种群的多样性,有利于进化过程的发展;种群的整体替换有利于加快种群的更新;Metropolis 抽样过程在增加局部搜索能力的同时赋予搜索过程避免局部极小的能力,而且这种能力是可控的,同时这种多点搜索很容易并行化处理。需要指出的是,由于计算机技术的高速发展,改进 GA 的优化过程完全可以加快,并且相对 NEH 方法其优化质量上的改进完全弥补了时间性能上一定程度的增加,当然进一步提高优化效率(如采用并行计算技术等)也是需深入的工作。

4.4 多目标 Flow Shop 调度的遗传算法

4.4.1 引言

鉴于大量实际工程问题属于多目标优化问题,因此多目标优化理论与方法一直是理论界和工程界共同关注的重要研究课题,生产调度领域也不例外。

考虑 n 个目标的最大化问题，即

$$\max \quad f_1(x), f_2(x), \cdots, f_n(x) \tag{4-4-1}$$

其中，$f_i(\cdot)$为优化目标。

定义 4.4.1 对于多目标最大化问题，称解 y 支配(dominate)解 x，即 y 相对于 x 有优势，若 $\forall i: f_i(x) \leqslant f_i(y)$ 且 $\exists j: f_j(x) < f_j(y)$。或者说，相对于解 y，解 x 是受支配的。

定义 4.4.2 对于多目标最大化问题，称解 x 为非受支配(non-dominated)解或 Pareto 最优解，若解空间中任意解均不能够支配解 x。

图 4.4.1 是两个目标下的最大化问题的受支配解和非受支配解的示意图，其中实心圆为非受支配解，空心圆为受支配解。鉴于 Pareto 最优解在目标空间(以目标优化为指标)中的位置，所有 Pareto 最优解的集合称为 Pareto 边界(frontier)。

需要指出的是，多目标优化的各目标之间往往存在冲突，并且 Pareto 最优解往往很多，而决策者往往希望优化算法能够给出多个多目标最优解以供选择，进而折中考虑各目标或侧重某一目标。由于遗传算法的种群并行搜索特点，GA 的进化结果是一组解，因此遗传算法已成为多目标优化的重要工具。但目前仍存在大量问题有待完善，譬如如何保证 GA 所得最终种群在多目标最优意义下具有分散性。

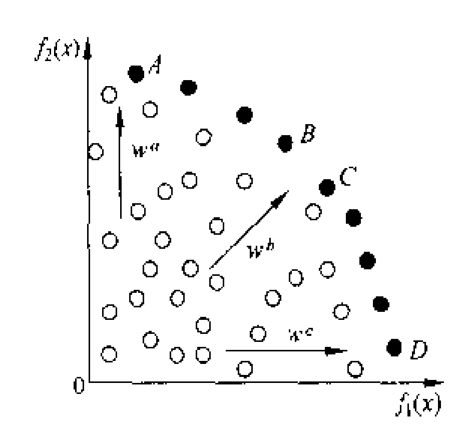


图 4.4.1 双目标最大化意义下的受支配解和非受支配解的示意图

目标加权是最常用的处理手段，如式(4-4-2)所示，如此可将一个多目标优化问题转化成一个单目标优化问题，并且 $f(x)$ 可作为遗传算法的适配值函数。

$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x) \tag{4-4-2}$$

其中，w_i 为权系数，且 $w_i \geqslant 0$，$i=1,2,\cdots,n$，$\sum_{i=1}^{n} w_i = 1$。

显然，给定一组权系数，优化过程将朝 Pareto 边界的一个方向进行，而不能够得到其他方向上的多目标最优解。譬如，令 $w_1 = w_2 = 0.5$，则算法最终将得到解 C，而得不到 A 和 D。因此，许多学者采用随机权系数，甚至是可变的。

此外，基于 GA 的优化结构，许多学者将算法的进化种群分解成若干子种群，各子种群分别用不同的权系数进行不同方向上的搜索。下面介绍 Ishibuchi 和 Murata (1998)提出的一种多目标遗传混合算法。

4.4.2 多目标遗传算法

本节介绍多目标最大化的遗传算法设计，强调多目标优化的特殊性。

1. 选择操作

在每次选择两个父代个体进行交叉操作时，首先由式(4-4-3)产生一组随机权系数，即

$$w_i = \xi_i / \sum_{j=1}^{n} \xi_j, \qquad i = 1,2,\cdots,n \tag{4-4-3}$$

其中，ξ_i 为非负随机数。

进而，由式(4-4-2)得到各个体的适配值。然后，通过线性变换可得到各个体的选择概率，如式(4-4-4)所示：

$$p_s = [f(x) - f_{\min}] / \sum_x [f(x) - f_{\min}] \tag{4-4-4}$$

其中，$f_{\min}$表示当前种群中最小的适配值。

于是，按照轮盘赌方式就可以进行个体的选择操作。对于所选的个体，通过交叉操作可得到一个新个体，然后对新个体进行变异操作，之后再进行局部搜索过程以最大化当前权系数下的适配值。显然，多次重复上述“选择一交叉一变异一局部搜索”过程，由于权系数的随机性，算法将得到多个不同权系数下的优良解，进而保证种群中个体对应搜索方向的多样性，如图4.4.2所示。

图 4.4.2 多方向搜索

2. 局部搜索过程

局部搜索过程是对“选择一交叉一变异”得到的解的趋向性搜索，以进一步提高个体的对应于加权目标的适配值。

[**局部搜索过程**]：

步骤 1：给定一个初始解 x，即遗传操作得到的解。

步骤 2：由当前解 x 的邻域产生一个解 y。

步骤 3：如果 $f(y)>f(x)$，则令 $x=y$；否则保持当前解不变。

步骤 4：如果比较了当前解的所有邻域解，则结束算法；否则返回步骤 2。

可见，上述算法是一个局部贪婪性搜索过程，可增强 GA 的局部搜索能力，当然也可以采用其他机制进行改进。然而，对于大规模问题，上述算法终止条件往往将导致很长的搜索过程。譬如，对于 20 个工件的置换排列，若以 SWAP 为邻域结构，则

每个解有 $C_{20}^{2}=190$ 个邻域解，从而整个多目标优化的搜索过程中局部搜索将占有太大的比重。因此，为了强调基于遗传操作的搜索，必须修改上述局部搜索的终止条件，譬如仅对当前解的 K 个邻域解的比较，而非所有邻域解。

3. 保优策略

为了保证不遗失非受支配解，每代进化过程后，旧的种群将被局部搜索过程得到的新种群所替代，同时非受支配解的暂定集合将由新的当前种群来更新。也就是说，如果当前种群中的一个解不受当前种群和非支配解的暂定集合中的其他任一解支配，那么这个解将加入暂定集合，同时删除暂定集合中受这个新增加解支配的所有解。另外，为了强调对暂定集合中非受支配解的改进，在上述对当前种群的"选择—交叉—变异—局部搜索"的同时，从暂定集合中随机选取若干非受支配解直接进行局部搜索，搜索的方向则采用选取该解父代个体时产生的权系数，若该解(譬如初始进化解)没有父代个体则采用随机权系数。这种保优策略可以用图 4.4.3 来描述，它包括 4 个过程，即当前种群对非受支配解的暂定集合的更新过程，当前种群个体的"选择—交叉—变异—局部搜索"过程，暂定集合中若干非受支配解的局部搜索过程以及下一代种群的构成。

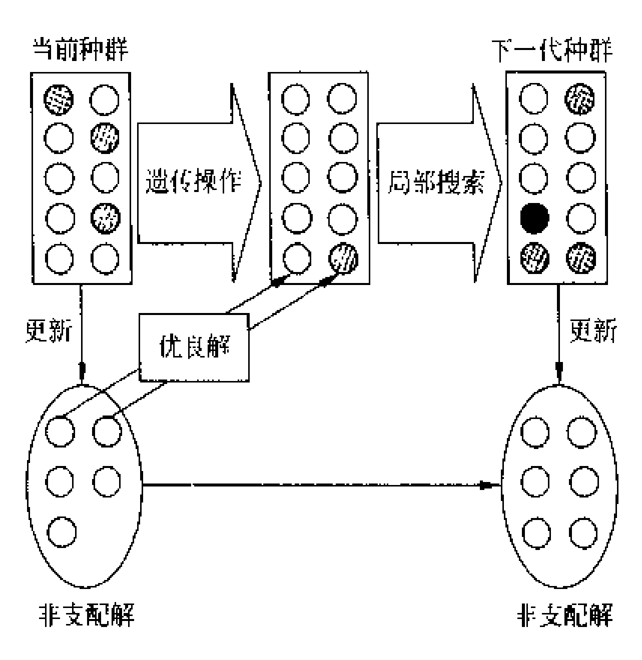


图 4.4.3 多目标混合遗传算法的每一代进化过程

4. 多目标混合遗传算法流程

基于上述描述，多目标优化的遗传算法可采用如下流程(其中 N_p 为种群规模，N_e 为保优解的规模)。由于算法中局部搜索的嵌入，以下称该 GA 为多目标混合遗传算法。

[**多目标混合遗传算法**]:

步骤1:随机产生 N_p 个初始个体形成初始种群。

步骤2:评价当前种群中各个体的适配值,更新非受支配解的暂定集合。

步骤3:重复如下步骤来选择(N_p-N_e)对父代种群。

(3.1) 按公式(4-4-3)产生随机权系数。

(3.2) 按公式(4-4-4)采用轮盘赌方式选取一对父代个体。

步骤4:对步骤3选取的(N_p-N_e)对父代个体执行交叉操作来得到(N_p-N_e)个新个体,然后对它们执行变异操作。

步骤5:从暂定集合中随机选取 N_e 个非受支配解,然后对步骤4变异操作后得到的(N_p-N_e)个解以及所选的 N_e 个非受支配解执行局部搜索过程得到 N_p 个新解。

步骤6:用步骤5得到的 N_p 个解替代原先种群成为新的当前种群。

步骤7:若算法终止准则满足则结束算法,否则转步骤2。

4.4.3 多目标 Flow Shop 调度的优化

对于多目标置换 Flow Shop 调度的求解,4.4.2 节遗传算法仍可采用工件置换排列作为编码,因此交叉、变异和局部搜索操作可采用求解单目标优化的任何操作。譬如,Ishibuchi 等(1998)采用两点交叉,即首先随机产生两个不同交叉点,后代个体首先继承父代个体1的两交叉点位置以外的基因片段,然后将父代个体2经删除父代个体1的两交叉点位置以外的基因的剩余基因片段填入后代个体的两交叉位置之间。如图 4.4.4 所示。同时,采用前插入操作作为变异和局部搜索的邻域结构,即首先随机确定两个变异位置,然后将后面位置上的基因插入到前面位置基因之前。显然,此时每一解存在 $(n-1)^2$ 个邻域解。

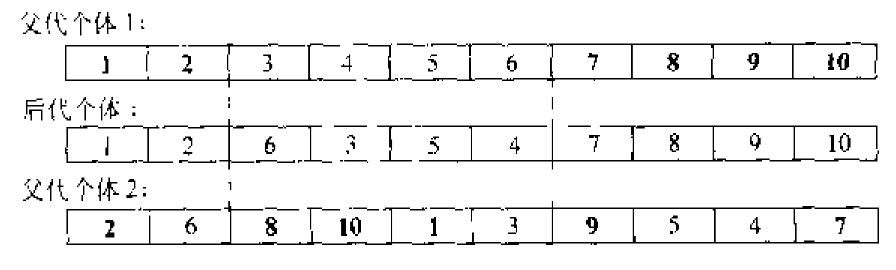


图 4.4.4 两点交叉示意图

下面结合具体数据解释多目标混合遗传算法的优化过程及其性能。

假定加工时间为区间[1, 99]的随机整数,然后随机给定一种所有工件的排列,并计算各工件的完成时间 c_i,进而设置各工件的交货期为 $d_i=c_i+\rho$,其中 ρ 为区间[-100,100]的随机整数。譬如,对于10工件5机器调度问题,其加工数据见

表 4.4.1　10 工件 5 机器置换 Flow Shop 问题的加工数据

工　件	1	2	3	4	5	6	7	8	9	10
交货期	674	396	431	369	626	597	790	437	656	780
机器 1 上加工时间	32	1	61	42	62	61	3	97	26	9
机器 2 上加工时间	21	27	87	45	59	24	71	34	20	28
机器 3 上加工时间	10	42	66	75	41	24	3	36	85	74
机器 4 上加工时间	51	19	23	85	86	81	93	31	75	23
机器 5 上加工时间	33	45	58	97	91	85	30	38	17	51

表 4.4.1。

考虑以最小化最大完成时间和最大拖后时间为目标的 n 个工件和 m 台机器的置换 Flow Shop 调度问题。鉴于最大完成时间的方差相对最大拖后时间要小，因此令目标 $f_1(x)$ 为 5 倍最大完成时间，目标 $f_2(x)$ 为 2 倍最大拖后时间。进而，适配值函数设置为

$$f(x)=-w_1 f_1(x)-w_2 f_2(x)$$

混合遗传算法参数选取如下：$N_p=20$，交叉概率 0.9，变异概率 0.3，$N_e=3$，$K=2$，整个算法的终止准则为 10 000 次个体评价（由于每代遗传操作评价 20 个解，每代进化中的局部搜索操作对解的评价次数不一定，因此算法实际运行的总进化代数也不一定，譬如文献用了 158 代进化）。多目标混合遗传算法的优化流程可以解释如下。

算法步骤 1 和 2　首先随机产生 20 个初始个体构成初始种群，其中的非受支配解构成暂定集合，如图 4.4.5 所示。

算法步骤 3　随机选取 17 对父代个体，同时产生 17 组权系数，譬如某一对父代个体对应的权系数为(0.503，0.497)，如图 4.4.6 所示。

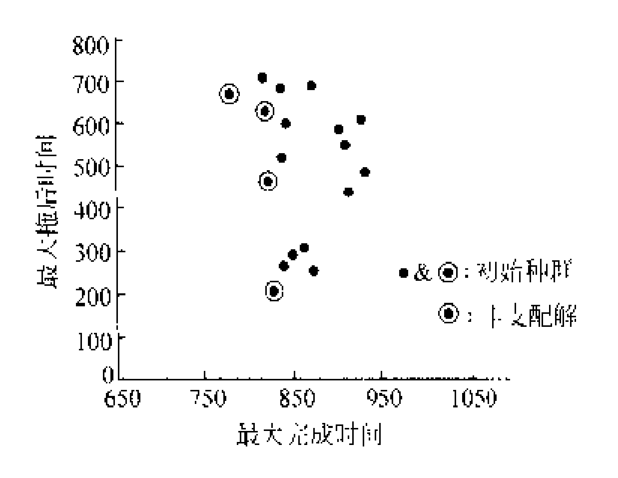


图 4.4.5　初始种群和初始非受支配解的暂定集合

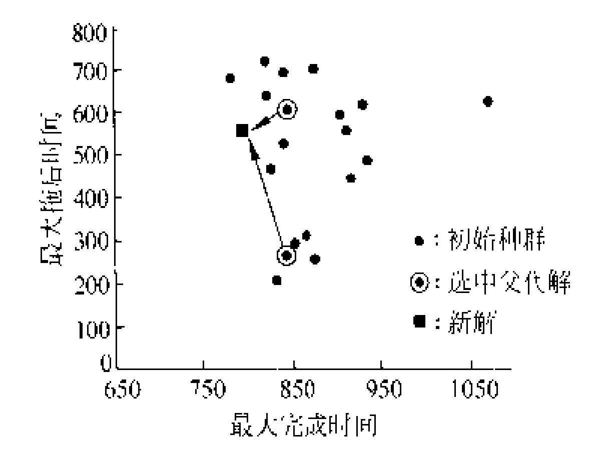


图 4.4.6　父代的选择及遗传操作产生的后代

算法步骤 4　对 17 对父代个体依次利用两点交叉和前插入变异操作得到 17 个新个体。

算法步骤 5　在暂定集合中随机选取 3 个非受支配个体与步骤 4 得到的 17 个新个体一起进行以前插入操作为邻域结构的局部搜索操作，从而得到 20 个新个体，进而它们在步骤 6 构成新的当前种群。局部搜索操作前后的 20 个个体如图 4.4.7 所示，可见局部操作对解性能的提高。然后，算法返回步骤 2，并重复上述过程直到个体的总评价次数达到 10 000。

上述多目标混合遗传算法达到的最终种群如图 4.4.8 所示；最终非受支配解如图 4.4.9 所示。

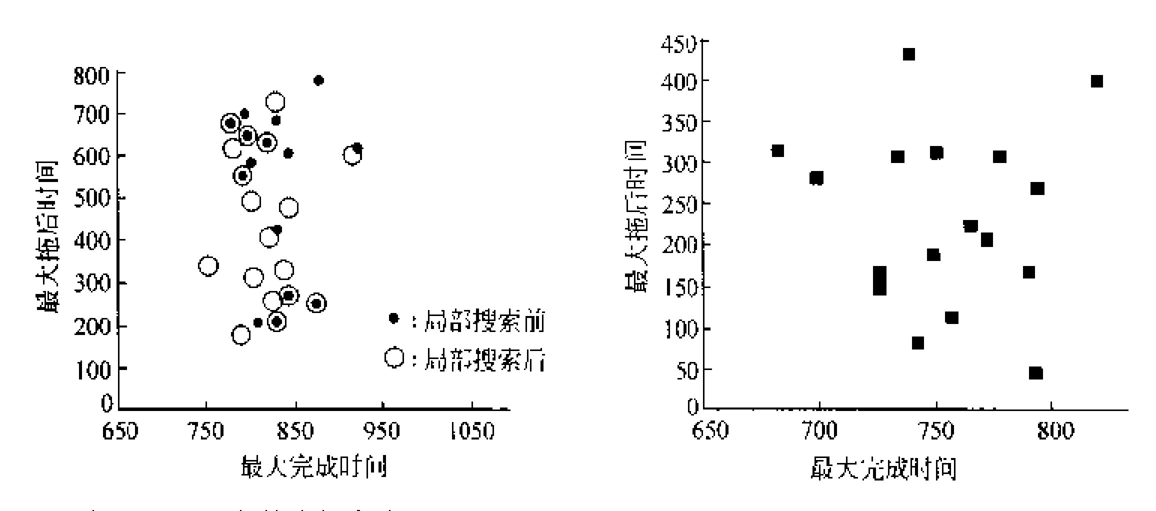


图 4.4.7　局部搜索操作前后的种群　　　图 4.4.8　多目标混合遗传算法得到的最终种群

由于解空间大小为 10!，故 Ishibuchi 等(1998)用穷举法验证了所得解在多目标意义下的最优性。需要指出的是，如果局部搜索的终止条件采用“对当前解邻域的完全列举”，则仿真结果没有得到图 4.4.9 中的所有 Pareto 最优解，因此在相同总计算量下，将计算量过分分配给局部搜索过程不是明智的选择。

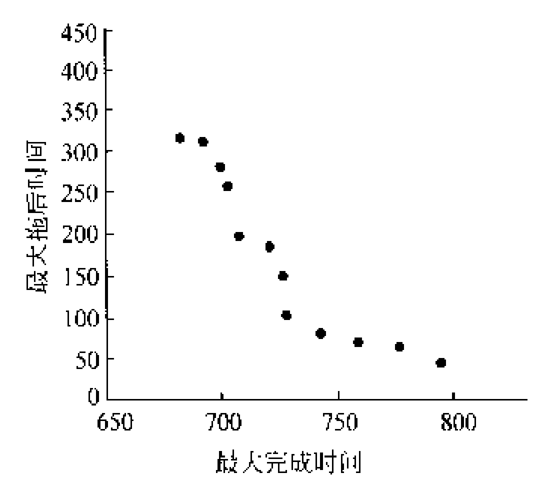


图 4.4.9　多目标混合遗传得到的非受支配解

另外，矢量评价遗传算法(vector evaluated GA, VEGA)的结果(如图4.4.10)，以及定常权系数GA(constant weight GA, CWGA)的结果(如图4.4.11)表明，它们均没有得到所有的非受支配解，因此上述多目标混合遗传算法是有效的。

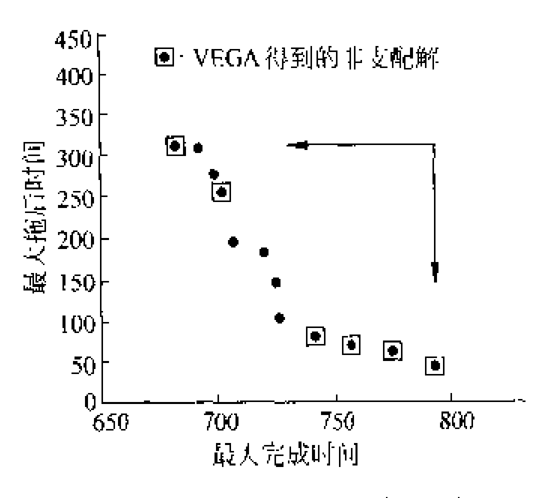


图4.4.10 VEGA的非受支配解

图4.4.11 CWGA得到的非受支配解

最后需要强调的是，上述多目标混合遗传算法具有通用性，不仅可应用于不同的优化问题(只需对遗传操作和局部搜索作调整)，而且可应用于多个不同的优化目标。

4.5 一类批量可变Flow Shop调度的遗传算法

4.5.1 问题描述

柔性流水线调度中同一工件往往包含若干数量，而且这些相同工件可以分成若干子批量进行加工，甚至加工子批量大小是可变的。本节考虑一类批量可变Flow Shop流水线调度问题。具体而言，假定所有机器呈流水线排列，各工件的加工顺序相同，在每一时刻，各机器只能加工一类工种的一个工件，且各工件只能在某一台机器上加工，给定加工时间、运输时间和工件加工设置时间(与加工顺序有关)，要求确定使最终加工完毕时间最小的各类工种的批量分割，以及所有子批量的最优排序。

显然，该问题中的批量和排序是相互制约的。Karmarkar等(1985)利用排队论模型证明，确定批量对流水时间和等待时间的影响实质上是寻找最优批量的路径优化；同时指出，批量与流水时间存在U型关系，即过大或过小的批量都会导致较长的流水时间。在此假定有N类工种，M台机器，每类工种的总批量均为L，且机器间缓冲区容量为无限。若以每类工种的每个工件为子批量，则批量总数为$N\times L$，解空间容量为$(N\times L)!$，不仅搜索效率低，而且由于过多的子批量数将导致设置和运输时间的增加，搜索空间中得到的最优解不一定是问题的全局最优解。若将每类工种视为一整体，则解空间容量为$N!$，此时较大批量占有机器会使前方机器长时间处于“饥

渴"状态，从而降低工作效率，并加大后方工件的等待时间。由于设置时间同排序有关，最优排序的搜索空间又同批量分割有关，故确定批量分割和排序的最优组合就成为解决问题的关键。下面，我们来设计求解该问题的改进遗传算法。

4.5.2 改进遗传算法

为了较好地解决该调度问题，本节设计一种结合遗传算法和模拟退火的混合优化策略。

1. 编码策略

借鉴压缩技术，可设计一种批量大小和排序的动态联合策略，以动态缩小搜索范围，即把同类工件合并为一个子批量(以下简称 JLS 操作)。譬如，3 类工种 A，B，C，各含 4 个工件，假定初始排序为 A(1)B(1)C(1)C(1)B(1)B(1)C(1)C(1)B(1)A(1)A(1)A(1)，其子批量数为 12，经 JLS 操作后的排序为 ABC(2)B(2)C(2)BA(3)，子批量数为 7。上述排序中括号内的数字表示各子批量所含的工件数。

JLS 操作本质上是合并操作。因此，在算法初始阶段以每个工件作为一个子批量，并在相应的解空间中产生初始种群。在进化过程中，一旦确定相应子批量数下的最优排列，就利用 JLS 操作来减小批量数，搜索空间也作相应变化。

2. 保优技术

JLS 操作对解空间的动态缩小，势必导致一些状态的不可返回。因此，可采用保优技术，随时保留搜索进程中发现的最佳状态。

3. 重升温技术

由于传统 SA 只有退温过程，高温时能够避免陷入局部极小，但低温时其突跳概率几乎为零。为了配合 JLS 操作，改善各阶段的搜索能力，可采用动态调温技术，在保留常规 SA 降温操作的同时，在每次 JLS 操作后适当提高温度。

4. 交叉操作

交叉操作采用简单易实现的 Non-ABEL 操作，以快速产生后代。假定父串分别为 a，b，则子串 c、d 分别为 $c[i]=a[b[i]]$，$d[i]=b[a[i]]$，$i=1,2,\cdots,n$。

5. 变异和 SA 状态产生函数

均采用互换操作。

6. 双阈值准则

为了使算法具有较好的优化性能，设置如下双阈值准则。

(1) 利用阈值 $STEP1$ 确定各子批量数下算法性能的改变情况，即若连续 $STEP1$ 代进化不能改善性能，则认为已得到该子批量数下的最优解。

(2) 利用阈值 $STEP2$ 确定 JLS 操作引起算法性能的变化，即若连续 $STEP2$ 次 JLS 操作没有改善优化性能，则认为已得到问题的最优解，终止算法。

7. 混合优化算法

[一类批量可变 Flow Shop 的改进遗传算法，简称 GSJLS]：

步骤 1：算法初始化：给定初温 t_0、子批量数、种群数 P_{size}、阈值 $STEP1$ 和 $STEP2$，令 $t=t_0$，$q=0$，$opt1=opt2=\infty$。

步骤 2：随机初始化种群，产生 P_{size} 个 1～K 的排列(K 为子批量数)，令 $p=0$。

步骤 3：选择种群中的两个最优个体以交叉概率进行交叉，产生两个子个体，保留四者中的两个优良个体。

步骤 4：对种群中的所有个体以变异概率进行变异。

步骤 5：对种群中的各个体进行 SA 操作：

(5.1) 由状态产生函数产生新个体并确定其目标值；

(5.2) 计算新、旧个体的目标函数差值 Δ；

(5.3) 以概率 $\min[1,\exp(-\Delta/t)]$接受新个体作为当前个体。

步骤 6：确定种群中的最优解，将其存入 $opt3$。

步骤 7：若 $opt1\leqslant opt3$，则令 $p=p+1$；否则，令 $opt1=opt3$，$p=0$。

步骤 8：若 $p<STEP1$，则 $t=\lambda\cdot t$ 并转步骤 3；否则，转步骤 9。

步骤 9：若 $opt2\leqslant opt1$，则令 $q=q+1$；否则，令 $opt2=opt1$，$q=0$。

步骤 10：若 $q<STEP2$，进行 JLS 操作，并确定新子批量数，令 $t=t_0\cdot\lambda^q$ 进行升温，然后转步骤 2；否则，转步骤 11。

步骤 11：输出最优解 $opt2$，计算时间，子批量数以及各子批量的排列。

通过分析可知，上述算法具有如下的优点：

(1) JLS 操作降低子批量总数，可动态缩小搜索范围，提高搜索效率；

(2) GA 群体并行搜索和 SA 概率突跳相结合，可加强搜索能力；

(3) 保优操作避免遗失优化中已搜索到的优良解；

(4) 重升温操作增大算法在批量数发生变化后的概率突跳能力；

(5) 双阈值策略使算法具有较好的优化性能。

4.5.3 仿真结果和分析

针对 5 类工种、5 台机器的问题，假定各工种均包含 10 个工件，加工时间为[0.1,1.1]的随机数，设置时间为[5,25]的随机数且与排序有关，运输时间取为[5,10]的随机数。为了研究问题对批量和排列联合依赖性，将上节给出的算法与以下两种方案作比较。

方案 1(简称 GSNJLS1)：每类工种视为一批量，搜索空间为 $N!$。

方案 2(简称 GSNJLS2)：各工件视为一批量，搜索空间为$(N\times L)!$。

算法参数选取如下：种群数 12、交叉概率 0.95，变异概率 0.01，初温 10，退温速

率 0.9，$STEP1$ 取 100，$STEP2$ 取 20，GSNJLS1 和 GSNJLS2 策略中阈值分别取为 100 和 500。

利用 Pentium586/120 计算机，以 Borland C++ 为环境，在上述参数下对各算法作 20 次随机仿真，仿真统计结果见表 4.5.1。

表 4.5.1 各算法仿真结果统计

算　法	最优值	波动率/%	平均搜索时间/s	最优子批量数
GSJLS	**140.32**	**5.92**	**7.11**	**13**
GSNJLS1	153.25	0	0.11	5
GSNJLS2	193.79	27.55	18.98	50

由仿真结果，可得到如下的一些结论：

GSNJLS1 策略在较小批量数对应的解空间中优化，尽管能够快速搜索到该批量数下的最优排列，但优化性能很差。原因是，子批量过大的工种占有机器，既导致前方机器长时间处于"饥渴"状态，也导致后方工件的长时间等待。因此，所得到的解并不是原问题的最优解。

GSNJLS2 策略在较大批量数对应的解空间中优化，搜索时间长，且优化性能差。原因是，子批量数过多导致过多的加工设置和运输时间。因此，得到的解也不是原问题的最优解。

改进遗传算法(GSJLS)联合考虑问题中的批量和排列，具有最佳优化性能，且时间性能和波动性能也较满意。因此可以认为，它是处理这类对批量分割与排序同时存在依赖性的问题的可行和有效算法。

4.6 模糊 Flow Shop 调度及其遗传算法

本节首先介绍模糊交货期下的 Flow Shop 调度及其遗传算法，进而描述模糊交货期框架下的其他调度指标，然后介绍模糊加工时间下的 Flow Shop 调度。

4.6.1 模糊交货期下 Flow Shop 调度的遗传算法

1. 问题描述

令 c_j 和 d_j 分别为工件 j 的加工完成时间和交货期，若调度问题以拖后工件数为指标，则显然 $c_j \leqslant d_j$ 是决策者最满意的情况，而 $c_j > d_j$ 是决策者不满意的情况(即有工件迟于其交货期完工)。由此，可按式(4-6-1)定义满意度。其示意图为图 4.6.1。

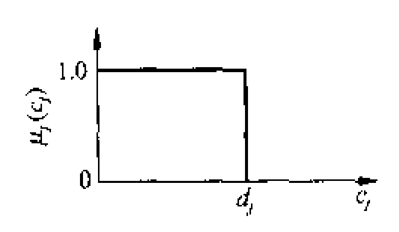


图 4.6.1 方波型满意度函数

$$\mu_j(c_j) = \begin{cases} 1, & c_j \leqslant d_j \\ 0, & c_j > d_j \end{cases} \tag{4-6-1}$$

若考虑模糊情况，即每个工件的交货期是 $\mathbf{R}^+$ 上的一个模糊集合，则可用其隶属度函数表示决策者对工件完工时间的满意度，而其形式可以多种多样，譬如图 4.6.2 的递减形式，图 4.6.3 的梯形形式（适用于 Just in-time 调度）。

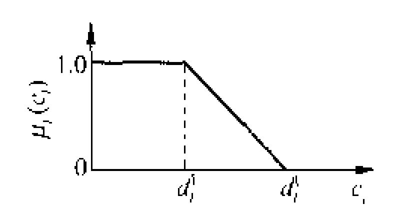


图 4.6.2　递减型满意度函数

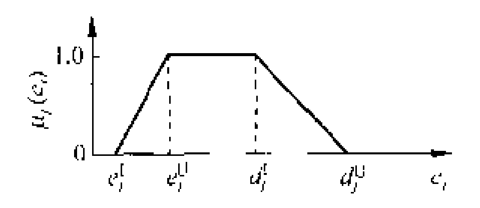


图 4.6.3　梯形型满意度函数

考虑模糊交货期下的常规置换 Flow Shop 调度，其数学描述同于 4.1.1 节，其中 $c_j = T_{j,m}$。如果各工件的模糊交货期已知，则该调度问题的求解过程仍是对所有工件排序的确定过程，而优化目标可以转化为最小满意度的最大化（如式(4-6-2)），或总满意度的最大化（如式(4-6-3)）。

$$\max\ f_{\min} = \min\{\mu_j(c_j),\quad j = 1,2,\cdots,n\} \tag{4-6-2}$$

$$\max\ f_{\text{sum}} = \sum_{j=1}^{n} \mu_j(c_j) \tag{4-6-3}$$

2. GA 设计

由于上述模糊调度问题除调度指标外均与常规问题相同，因此可以采用求解常规问题的任何遗传算法。在此介绍 Ishibuchi 等(1994)的一种方案。

令 N_p 为种群数，$P(t) = \{X_i^t,\ i = 1,2,\cdots,N_p\}$ 为第 t 代种群，其中 X 表示 n 个工件的置换排列，直接令式(4-6-2)或式(4-6-3)为适配值函数 $g(X)$，则 GA 流程如下。

［**模糊交货期下置换 Flow Shop 调度的遗传算法**］：

步骤 1：随机产生 N_p 个排列构成初始种群 $P(1)$。

步骤 2：利用基于线性变换的轮盘赌方式选择 $(N_p - 1)$ 对父代个体，选择概率如式(4-6-4)。

步骤 3：对每一对所选父代个体采用 4.4.3 节的两点交叉以交叉概率进行交叉，产生 $(N_p - 1)$ 个新个体，若某对父代个体未进行交叉，则随机选一个个体作为新个体。

步骤 4：以变异概率对步骤 3 得到的新个体进行变异操作。

步骤 5：将步骤 4 得到的 $(N_p - 1)$ 个个体和父代种群中的最优个体共同构成下一代种群。

步骤6：若算法终止准则满足，则结束算法；否则返回步骤2。

$$p(X_i^t)=[g(X_i^t)-g_{\min}(P(t))]\Big/\sum_{j=1}^{n}[g(X_j^t)-g_{\min}(P(t))] \quad (4\text{-}6\text{-}4)$$

其中，$g_{\min}(P(t))=\min\{g(X_i^t),X_i^t\in P(t)\}$。

3. 仿真结果

首先考虑如图4.6.2所示的递减满意度情况，即

$$\begin{cases}1, & c_j\leqslant d_j^L\\ 1-(c_j-d_j^L)/(d_j^U-d_j^L), & d_j^L<c_j\leqslant d_j^U\\ 0, & c_j>d_j^U\end{cases} \quad (4\text{-}6\text{-}5)$$

分别考虑100个10工件、20工件、50工件的问题，机器数统一为10台，各操作的加工时间为区间[1, 99]内随机整数。其中，d_j^L 和 d_j^U 按如下方式给出。

[d_j^L 和 d_j^U 的确定]：

步骤1：对每一测试问题，随机产生一种工件排列 X。

步骤2：计算该排列下各工件的完工时间 $c_j(X)$。

步骤3：令 $d_j^L=c_j(X)-\varepsilon, d_j^U=c_j(X)+\varepsilon$。

于是，式(4-6-5)可改写为下式：

$$\begin{cases}1, & c_j\leqslant c_j(X)-\varepsilon\\ 1-[c_j-c_j(X)+\varepsilon]/2\varepsilon, & c_j(X)-\varepsilon<c_j\leqslant c_j(X)+\varepsilon\\ 0, & c_j(X)+\varepsilon<c_j\end{cases} \quad (4\text{-}6\text{-}6)$$

若最终排列与 X 相同，则对所有工件 $\mu_j(c_j)=0.5$。这对于指标(4-6-2)显然是最优的，而对于指标(4-6-3)却不一定最优。

选取遗传算法参数：种群数50，交叉概率0.9，变异概率0.3，终止条件为2000代固定进化。优化结果见表4.6.1。

表4.6.1 模糊交货期下置换Flow Shop调度的GA性能

调度目标	工件数	10	20	50
指标(式(4-6-2))	最小满意度	0.500	0.000	0.000
	平均满意度	0.500	0.485	0.499
指标(式(4-6-3))	总满意度	8.376	17.797	46.693
	平均满意度	0.838	0.890	0.934

显然，GA较好地解决了指标(4-6-3)下的模糊交货期Flow Shop调度问题，但求解指标(4-6-2)下的较大规模问题性能较差。其原因在于：首先，若存在一个工件的满意度为零，则指标(4-6-2)将为零；其次，进一步的大量数值研究表明，该类问题对

应的适配值函数值相同，因此 GA 成为随机盲目搜索(Ishibuchi 等，1994)。由此，将图 4.6.3 对应的满意度函数修改成如图 4.6.4 所示的形式，相应的计算公式为式(4-6-7)。相同数据下，$a=1$ 时的仿真结果见表 4.6.2。

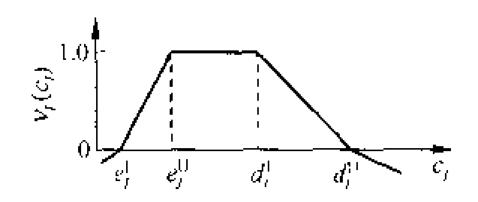


图 4.6.4 修改的梯形型满意度函数

$$v_j(c_j)=\begin{cases}a\cdot(c_j-e_j^L), & c_j<e_j^L\\ \mu_j(c_j), & e_j^L\leqslant c_j\leqslant d_j^U\\ a\cdot(d_j^L-c_j), & d_j^U<c_j\end{cases} \tag{4-6-7}$$

其中，a 为一个正系数。

表 4.6.2 修改满意度函数下的 GA 性能

工件数	10	20	50
最小满意度	0.499	0.399	−69.030
平均满意度	0.561	0.621	−10.409

由表 4.6.2 可见，20 工件问题的结果改善了，但 50 工件问题出现了负指标值。由于递减满意度函数下指标(4-6-2)与工件的拖后时间相关，因此可设计一个规则来构造调度，即按工件交货期大小递增次序加工各工件。由于模糊交货期问题没有一个确定性的交货期值，因此采用 d_j^L 的递增次序为参考进行工件排列。表 4.6.3 为该规则性方法的结果，表 4.6.4 为规则性方法与 GA 结合的结果(即初始种群由规则性方法得到的一个初始解和 N_p-1 个随机解构成)。仿真结果表明，规则性方法的结果好于纯 GA 方法，而 GA 与规则性方法的结合能够得到更好的解。这又一次说明了算法混合的有效性。

表 4.6.3 规则性方法的性能

工件数	10	20	50
最小满意度	0.343	0.296	0.205
平均满意度	0.507	0.515	0.501

表 4.6.4 GA 与规则性方法结合的性能

工件数	10	20	50
最小满意度	0.506	0.472	0.367
平均满意度	0.500	0.557	0.529

4.6.2 模糊交货期下的其他指标

本节给出模糊交货期的其他调度指标的描述,进而可将 GA 应用到其他指标的模糊交货期调度问题,甚至是多目标优化问题。

1. 拖后工件数

若采用如图 4.6.1 所示的方波型满意度函数,则指标(4-6-3)转化为式(4-6-8),也即最小化拖后工件数指标,即

$$\max f_{sum} = \sum_{j=1,c_j \leqslant d_j}^{n} 1 \quad 或 \quad \min f = \sum_{j=1,d_j<c_j}^{n} 1 \tag{4-6-8}$$

2. 最大拖后时间和总拖后时间

由定义知,拖后时间 $T_j = \max\{0, c_j - d_j\} = (c_j - d_j)^+$。考虑图 4.6.2 所示的递减型满意度函数,假定所有工件满足 $c_j < d_j^U$(譬如设置 d_j^U 足够大),即满意度为正值,则指标(4-6-2)转化如下:

$$\max f_{min} = \{1 - (c_j - d_j^L)^+ /(d_j^U - d_j^L)\}$$

或

$$\min f = \max\{(c_j - d_j^L)^+ /(d_j^U - d_j^L)\} \tag{4-6-9}$$

进一步,假定 $d_j^U - d_j^L$ 对所有工件均相同,则式(4-6-9)简化为最小化 $f = \max\{(c_j - d_j^L)^+\}$。若取 $d_j^L = d_j$,则上述指标就成为最小化最大拖后时间。类似地,可知最小化总拖后时间指标是指标(4-6-3)的特例。

3. 超前和拖后惩罚

由定义知,超前时间 $E_j = \max\{0, d_j - c_j\} = (d_j - c_j)^-$。超前拖后指标通常为

$$\min ET_{max} = \max\{\alpha_j E_j + \beta_j T_j\} \tag{4-6-10}$$

考虑三角满意度函数,即满足 $e_j^U = d_j^L$ 的图 4.6.2 情况,其计算公式如下:

$$\mu_j(c_j) = \begin{cases} 0, & c_j \leqslant e_j^L \\ 1 - (e_j^U - c_j)/(e_j^U - e_j^L), & e_j^L < c_j \leqslant e_j^U \\ 1 - (c_j - d_j^L)/(d_j^U - d_j^L), & d_j^L < c_j \leqslant d_j^U \\ 0, & d_j^U < c_j \end{cases} \tag{4-6-11}$$

令 $\gamma_j = 1/(e_j^U - e_j^L)$,$\delta_j = 1/(d_j^U - d_j^L)$,假定对所有工件 $e_j^L < c_j < d_j^U$,$e_j^U = d_j^L = d_j$,则式(4-6-11)可改写如下:

$$\begin{aligned} \mu_j(c_j) &= \begin{cases} 1 - \gamma_j(e_j^U - c_j), & c_j \leqslant e_j^U \\ 1 - \delta_j(c_j - d_j^L), & d_j^L < c_j \end{cases} \\ &= \begin{cases} 1 - \gamma_j(d_j^L - c_j), & d_j^L - c_j \geqslant 0 \\ 1 - \delta_j(c_j - d_j^L), & c_j - d_j^L > 0 \end{cases} \\ &= 1 - \gamma_j(d_j - c_j)^+ - \delta_j(c_j - d_j)^+ \end{aligned} \tag{4-6-12}$$

从而,指标(4-6-2)可写成

$$\max f_{\min} = \min\{1-\gamma_j(d_j-c_j)^- - \delta_j(c_j-d_j)^+\}$$

或

$$\min f = \max\{\gamma_j(d_j-c_j)^+ + \delta_j(c_j-d_j)^+\}$$

因此，超前拖后指标是指标(4-6-2)的一种特例。类似地，可知总超前拖后时间为指标(4-6-3)的特例。

4. 最大完成时间和总流经时间

最大完成时间，即 $C_{\max}=\max\{c_j\}$。考虑如下满意度计算公式：

$$\mu_j(c_j) = \begin{cases} 1 - c_j/d^{U}, & c_j < d^{U} \\ 0, & c_j \geq d^{U} \end{cases} \tag{4-6-13}$$

此时，指标(4-6-2)可改写为

$$\max f_{\min} = \min\{1 - c_j/d^{U}\}$$

或

$$\min f = \max\{c_j/d^{U}\}$$

也即 Makespan 指标。类似地，可知总流经时间是指标(4-6-3)的特例。

4.6.3 模糊加工时间下 Flow Shop 调度的遗传算法

考虑模糊加工时间下的置换 Flow Shop 调度问题，即 4.1.1 节的数学描述中各操作的加工时间为模糊值。令 $\tilde{t}_{i,j}$ 为工件 i 在机器 j 上的模糊加工时间，约定模糊加工时间 $\tilde{t}$ 为三角模糊数 TFN，用(u,t,v)表示，其隶属度函数如图 4.6.5 所示。

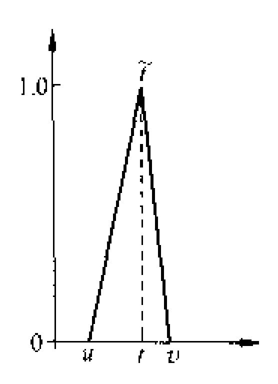


图 4.6.5 TFN 模糊数的隶属度函数

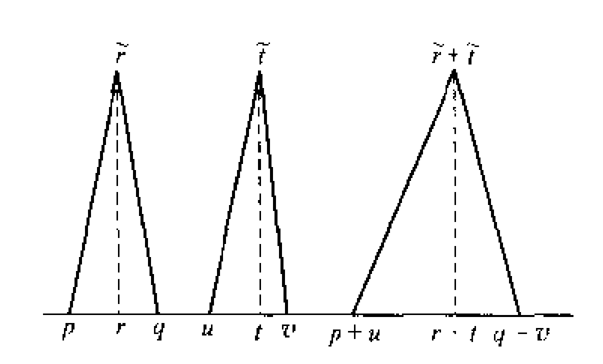


图 4.6.6 TFN 的加法运算

下面定义模糊加、模糊取大运算，用以确定各操作的完工时间。

1. TFN 的加法运算

累加计算将用于工件的完成时间。若两 TFN $\tilde{r}$ 和 $\tilde{t}$ 分别用(p,r,q)和(u,t,v)表示，则其和 $\tilde{r}+\tilde{t}$ 为$(p+u,r+t,q-v)$，如图 4.6.6 所示。

2. TFN 的取大运算

取大操作将用于确定工件的加工开始时间。记上述 TFN $\tilde{r}$ 和 $\tilde{t}$ 的隶属度函数为 $u_{\tilde{r}}$ 和 $u_{\tilde{t}}$，用 ∨ 表示取大操作，则

$$\tilde{r} \vee \tilde{t} = (p,r,q) \vee (u,t,v)$$

其隶属度函数为

$$u_{\tilde{i}\vee\tilde{t}}(z)=\sup_{z=x\vee y}\min\{u_{\tilde{i}}(x),u_{\tilde{t}}(y)\}$$

取大的结果如图 4.6.7 中阴影部分所示。当然，也可采用第 3 章中处理模糊 Job Shop 调度的近似方法。

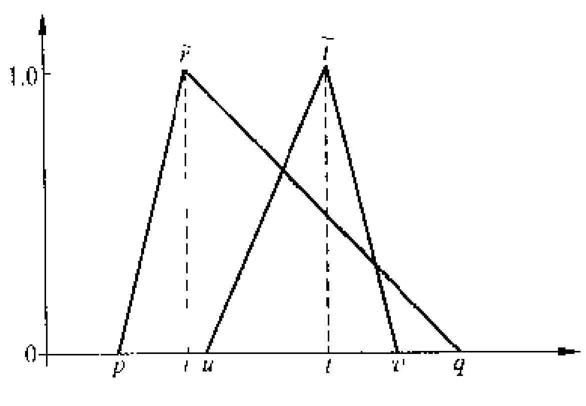


图 4.6.7　TFN 的取大运算

对于模糊加工时间的调度问题，由于工件完工时间 $\tilde{c}_i$ 的模糊性，拖后时间 $\tilde{T}_i=\max\{0,\tilde{c}_i-d_i\}$、超前时间 $\tilde{E}_i=\max\{0,d_i-\tilde{c}_i\}$、最大完成时间 $\tilde{C}_{\max}=\max\{\tilde{c}_i\}$ 等数据也将模糊化。进而，优化的调度指标也将模糊化。譬如最小化模糊 E/T 指标 $\min\max\{\alpha_i\tilde{E}_i+\beta_i\tilde{T}_i\}$，最小化最大完成时间 $\min\tilde{C}_{\max}$ 等。然而，一旦模糊运算和模糊调度指标确定，调度性能的好坏仍完全依赖于工件的排列，因而求解模糊加工时间下的 Flow Shop 等调度问题的遗传算法仍可采用前文介绍的设计方案，甚至是多模糊目标优化问题。在此不再介绍其具体的算法流程和仿真结果。

4.7　混合 Flow Shop 调度的遗传算法

4.7.1　问题描述

混合 Flow Shop 调度(hybrid flow shop scheduling problem，HFSP)是传统 Flow Shop 问题的一种推广。该问题可描述如下：n 个工件在流水线上进行 m 个阶段的加工，每一阶段至少有一台机器且至少有一个阶段存在多台机器，并且同一阶段上各机器的处理性能相同，在每一阶段各工件均要完成一道工序，各工件的每道工序可以在相应阶段上的任意一台机器上加工，已知工件各道工序的处理时间，要求确定所有工件的排序以及每一阶段上机器的分配情况，使得调度指标最小。图 4.7.1 给出了 HFSP 问题的一个图例，其中有 m 个阶段，每阶段有 M_i 台并行机。进一步，如果要求各阶段所有工件的加工优先顺序(即占有并行机的优先权)相同，则该问题称为置换 HFSP 调度。

以 Makespan 为指标的置换 HFSP 调度可用如下混合整数线性规划模型来描述：

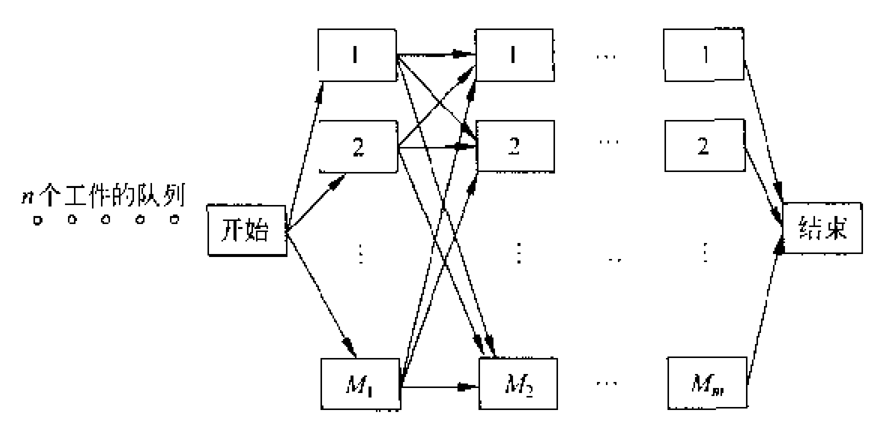


图 4.7.1　HFSP 调度问题图示

$$\min \max_{i=1,\cdots,n} \{e_{im}\} \tag{4-7-1}$$

$$\sum_{i=1}^{n} x_{il} = 1, \quad l = 1,2,\cdots,n \tag{4-7-2}$$

$$\sum_{l=1}^{n} x_{il} = 1, \quad i = 1,2,\cdots,n \tag{4-7-3}$$

$$\sum_{k=1}^{M_j} y_{ijk} = 1, \quad i = 1,\cdots,n; \quad j = 1,\cdots,m \tag{4-7-4}$$

$$e_{ij} = s_{ij} + t_{ij}, \quad i = 1,\cdots,n; \quad j = 1,\cdots,m \tag{4-7-5}$$

$$e_{ij} \leqslant s_{i(j+1)}, \quad i = 1,\cdots,n; \quad j = 1,\cdots,m \tag{4-7-6}$$

$$\sum_{i=1}^{n} x_{il} s_{ij} \leqslant \sum_{i=1}^{n} x_{i(l+1)} s_{ij}, \quad i,l = 1,\cdots,n; \quad j = 1,\cdots,m \tag{4-7-7}$$

$$\sum_{i=1}^{n} x_{il_1} y_{ijk} e_{ij} \leqslant \sum_{i=1}^{n} x_{il_2} y_{ijk} s_{ij} + \left(1 - \sum_{i=1}^{n} x_{il_2} y_{ijk} s_{ij}\right) \cdot L, \tag{4-7-8}$$

$$j = 1,\cdots,m; \quad l_1, l_2 = 1,\cdots,n; \quad l_1 \leqslant l_2; \quad k = 1,\cdots,M_j$$

其中，t_{ij} 表示工件 i 的第 j 道工序的加工时间；L 为足够大数；x_{il} 为 0-1 变量（若工件 i 被安排在第 l 个位置则取值为 1，否则取值为 0）；y_{ijk} 也为 0-1 变量（若工件 i 的第 j 道工序被分配在第 k 台机器上则取值为 1，否则取值为 0）；s_{ij} 表示工件 i 的第 j 道工序的开始加工时间；e_{ij} 则表示工件 i 的第 j 道工序的完成时间。式(4-7-1)表示调度指标为最大完成时间，即 Makespan；式(4-7-2)和式(4-7-3)保证调度为所有工件的一个完全排列；式(4-7-4)表示任何一个阶段每个工件只能由一台机器加工；式(4-7-5)表示同一阶段上工序完成时间和开始时间的关系；而式(4-7-6)则表示同一工件在进行下一道工序之前必须完成当前工序；式(4-7-7)表示调度排列中排位越前的工件开始处理时间越早；式(4-7-8)则表示同一阶段分配在同一机器上的工件，排位靠后的工件必须等排位靠前的工件加工完毕后才可进行。而设置足够大数 L 则是使式(4-7-8)当处于不同排位的工件不在同一机器上加工时也成立。

由于 HFSP 调度问题不仅要解决所有工件的排序，而且需要解决各阶段并行机的分配问题，因此其求解比 FSP 更为复杂和困难。鉴于 HFSP 问题有很强的工程背景，尤其在冶金和化工工业，该问题一直是调度领域关心的重要研究问题。譬如，Linn 等(1999)简单综述了多阶段 HFSP 的若干研究现状，但未涉及到智能优化算法的设计。Santos 等(1995)给出了求解 HFSP 最大完成时间的一个下界，可用于检验次优化方法的性能好坏。Brah 等给出了小规模 HFSP 问题的一种分支定界方法(Brah et al. 1991)和启发式方法(Brah et al, 1999)。Xiao 等(2000)提出了 HFSP 问题的一种遗传算法。总之，HFSP 调度的智能优化算法研究还不够完善，大量研究工作有待深入。下面介绍 HFSP 调度的遗传算法设计。

4.7.2 基于矩阵编码的遗传算法设计

本节介绍一般 HFSP 调度问题的一种基于矩阵编码的遗传算法设计，包括编码、种群初始化、交叉和变异，而适配值函数、选择、终止准则和算法参数则同于一般 GA 的设计。参见第 2 章有关内容。

1. 编码

对于上述 HFSP 调度问题，采用如下矩阵编码：

$$\mathbf{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{4-7-9}$$

其中，a_{ij} 为区间$(1,M_i+1)$上的一个实数，表示工件 j 的第 i 道工序在第$\lfloor a_{ij}\rfloor$台机器上加工($\lfloor\cdot\rfloor$表示取下整数)。若出现$\lfloor a_{ij}\rfloor=\lfloor a_{ik}\rfloor$，$j\neq k$，则表示多个工件的工序将在同一台机器上加工。此时，若 $i=1$，则可以约定按 a_{ij} 值由小到大确定各工件的加工顺序。若 $i>1$，则根据每个工件的前道工序的完成时间来确定其加工顺序，即前道工序先完成的先加工(FIFO)。若前道工序完成时间相同，则按 a_{ij} 值由小到大确定各工件的加工顺序。若 a_{ij} 值再相同，则随机确定加工顺序。如此可将机器分配和工件排列结合考虑。

进而，基于该矩阵编码可以构造用于遗传算法搜索的染色体，即一个 $n\times m+m-1$ 的字符串。该字符串包含编码矩阵的各行，不同行之间用 0 隔开，也即

$$[a_{11},a_{12},\cdots,a_{1n},0,a_{21},a_{22},\cdots,a_{2n},0,\cdots,0\cdots,a_{m1},a_{m2},\cdots,a_{mn}]$$

譬如，3 工件 3 阶段 HFSP 问题，各阶段分别包含 3,2,2 台机器，矩阵码为

$$\mathbf{A} = \begin{bmatrix} 2.2 & 2.4 & 1.8 \\ 1.5 & 2.1 & 2.3 \\ 1.1 & 2.4 & 1.3 \end{bmatrix}$$

则通过对该矩阵各元素取整后确定调度方案，即：工件 1 的 3 道工序分别在 3 个阶

段的第 2、1，1 台机器上加工；工件 2 的 3 道工序分别在 3 阶段的第 2，2，2 台机器上加工；工件 3 的 3 道工序分别在 3 个阶段的第 1，2，1 台机器上加工。至于各工件的加工顺序，对于第 1 道工序，工件 1 和工件 2 均在第 1 阶段的第 2 台机器上加工，而由于 2.2<2.4，因此先加工工件 1，后加工工件 2；对于第 2 道工序，工件 2 和 3 均在该阶段的第 2 台机器上加工，但要根据它们前道工序的完工时间来确定其先后关系。若前道工序完工时间相同，则由 2.1<2.3 知工件 2 先加工。相应的染色体为 [2.2，2.4，1.8，0，1.5，2.1，2.3，0，1.1，2.4，1.3]。

2. 初始种群

产生初始种群即产生一组矩阵，矩阵的各元素在其相应的区间内随机取值，进而再将这些矩阵转换为染色体群。

3. 交叉操作

显然，只要保证 a_{ij} 位于区间 $(1, M_i+1)$ 之内就能够保证染色体的合法性。因此，交叉操作可以在此原则上多样化设计。譬如，对于染色体的每一个子片段，即矩阵编码的每一行，随机确定某些位置的基因跟另一个染色体的相同位置上的基因互换。例如，若父代染色体为

$$A1 = [\underline{2.2,\ 2.4},\ 1.8,\ 0,\ 1.5,\ \underline{2.1},\ 2.3,\ 0,\ 1.1,\ \underline{2.4,\ 1.3}]$$

$$A2 = [\underline{1.1,\ 2.1},\ 2.8,\ 0,\ 1.8,\ \underline{1.1},\ 2.8,\ 0,\ 2.5,\ \underline{2.1,\ 2.1}]$$

则生成的后代染色体为

$$B1 = [\underline{1.1,\ 2.1},\ 1.8,\ 0,\ 1.5,\ \underline{1.1},\ 2.3,\ 0,\ 1.1,\ \underline{2.1,\ 2.1}]$$

$$B2 = [\underline{2.2,\ 2.4},\ 2.8,\ 0,\ 1.8,\ \underline{2.1},\ 2.8,\ 0,\ 2.5,\ \underline{2.4,\ 1.3}]$$

其中，下加"_"者表示互换基因。

4. 变异操作

类似交叉操作，变异也可采用分段方式，而对第 i 个片段的第 j 个基因，首先在集合 $\{-1, 1\}$ 中随机取值得到 K。若 $K=1$，则产生随机数 $r=\text{rand}(0, M_i+1-a_{ij})$；否则产生随机数 $r=\text{rand}(0, a_{ij}-1)$。进而令 $a_{ij}=a_{ij}+K\cdot r$ 来实现基因的变异。显然，这种操作能够保证 a_{ij} 位于区间 $(1, M_i-1)$ 之内，从而保证染色体的合法性。譬如，若父代染色体为

$$A1 = [2.2,\ 2.4,\ 1.8,\ 0,\ 1.5,\ 2.1,\ 2.3,\ 0,\ 1.1,\ 2.4,\ 1.3]$$

$$K = [1,\ 1,\ -1,\ 0,\ -1,\ 1,\ 1,\ 0,\ 1,\ -1,\ 1]$$

$$r = [1.5,\ 0.4,\ 0.7,\ 0,\ 0.3,\ 0.5,\ 0.2,\ 0,\ 1.1,\ 0.6,\ 0.9]$$

则后代染色体为

$$B = [3.7,\ 2.8,\ 1.1,\ 0,\ 1.2,\ 2.6,\ 2.5,\ 0,\ 2.2,\ 1.8,\ 2.2]$$

4.7.3 基于置换编码的遗传算法设计

显然，一旦给定所有工件进入流水线的顺序（即工件的排列）和各阶段上并行机

的分配方案，所有工件的各道工序的加工开始时间和完成时间就可确定，因此 HFSP 问题也可基于置换编码设计遗传算法。也即，类似于传统置换 FSP 问题，采用所有工件的置换排列作为遗传算法的染色体编码方式。譬如，若 $n=5$，$m=2$，$M_1=2$，$M_2=3$，则染色体(1 3 4 2 5)表示所有工件进入流水线的顺序为“工件 1—工件 3—工件 4—工件 2—工件 5”。

对于一般 HFSP 调度，可以采用 FIFO 的方式进行同一阶段上并行机的分配，即越先进入某一阶段的工件越早占有闲置的并行机，如果没有闲置的机器就必须等待，直到有闲置的机器。显然，一旦固定第 1 阶段的工件排列，后续各阶段的工件加工情况也就确定了。下面对此分配策略给予示例。

假定 HFSP 规模和工件排列同上，若工件 1 至工件 5 在第 1 和第 2 阶段上的加工时间分别为(30 51 18 74 9)和(35 70 22 62 14)，则两阶段上并行机的具体分配情况如下。

1. 第 1 阶段(该阶段有 2 台并行机)

(1) 首先将工件 1 和工件 3 分配给第 1 阶段的第 1 和第 2 台并行机，它们的加工完成时间分别为 30 和 18，也就是第 1 和第 2 台并行机的释放时间；

(2) 由于 18<30，因此在时刻 18，将工件 4 分配给第 2 台并行机，其加工完成时间和该机器的释放时间为 18+74=92。

(3) 由于 30<92，因此在时刻 30，将工件 2 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 30+51=81。

(4) 由于 81<92，因此在时刻 81，将工件 5 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 81+9=90。

2. 第 2 阶段(该阶段有 3 台并行机)

(1) 由于第 1 阶段最早完成的工件为工件 3、工件 1 和工件 2，则它们首先分别占有第 2 阶段的第 1、第 2 和第 3 台机器，其加工完成时间和相应机器的释放时间分别为 18+22=40，30+35=65，81+70=151。

(2) 由于 40<65<151，同时工件 5 早于工件 4 进入第 2 阶段，因此在时刻 40，将工件 5 分配给第 1 台并行机，加工完成后该机器的释放时间为 40+14=54。

(3) 由于 54<65<151，因此在时刻 54，将工件 4 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 54+62=116。

因此，工件 1 至工件 5 在第 1 阶段和第 2 阶段的加工完成时间分别为(30 81 18 92 90)和(65 151 40 116 54)；第 1 阶段第 1 台并行机依次加工工件 1，2 和 5，其释放时间为 90；第 1 阶段第 2 台并行机依次加工工件 3 和 4，其释放时间为 92；第 2 阶段第 1 台并行机依次加工工件 3，5 和 4，其释放时间为 116；第 2 阶段第 2 台并行机加工工件 1，其释放时间为 65；第 2 阶段第 3 台并行机加工工件 2，其释放时间为 151；整个 HFSP 调度问题的 Makespan 值为 151。

对于置换 HFSP 调度，每一阶段上所有工件加工优先顺序固定，即处于排列越前的工件具有越高的优先加工权、优先占有该阶段上的闲置并行机。下面对此分配策略给予示例。

假定 HFSP 规模、工件排列和各工件各道工序的加工时间同上，则两阶段上并行机的具体分配情况如下。

1. 第 1 阶段(该阶段有 2 台并行机)

(1) 首先将工件 1 和工件 3 分配给第 1 阶段的第 1 和第 2 台并行机，它们的加工完成时间分别为 30 和 18，也就是第 1 和第 2 台并行机的释放时间。

(2) 由于 18<30，因此在时刻 18，将工件 4 分配给第 2 台并行机，其加工完成时间和该机器的释放时间为 18+74=92。

(3) 由于 30<92，因此在时刻 30，将工件 2 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 30+51=81。

(4) 由于 81<92，因此在时刻 81，将工件 5 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 81+9=90。

2. 第 2 阶段(该阶段有 3 台并行机)

(1) 由于工件加工优先顺序的限制，首先将工件 1，3 和 4 分别分配给第 2 阶段的第 1，2 和 3 台并行机，其加工完成时间和相应并行机的释放时间分别为 30+35=65，18+22=40 和 92+62=154，它们也分别为第 1，2 和 3 台并行机的释放时间。

(2) 由于工件 2 的加工优先权高于工件 5，同时 40<65<154，因此在时刻 40，将工件 2 分配到第 2 台并行机，其加工完成时间和该并行机的释放时间为 40+70=110。

(3) 由于 65<110<154，因此在时刻 65，将工件 5 分配给第 1 台并行机，其加工完成时间和该机器的释放时间为 65+14=79。

因此，工件 1 至工件 5 在第 1 阶段和第 2 阶段的加工完成时间分别为(30 81 18 92 90)和(65 110 40 154 79)；第 1 阶段第 1 台并行机依次加工工件 1，2 和 5，其释放时间为 90；第 1 阶段第 2 台并行机依次加工工件 3 和 4，其释放时间为 92；第 2 阶段第 1 台并行机依次加工工件 1 和 5，其释放时间为 79；第 2 阶段第 2 台并行机加工工件 3 和 2，其释放时间为 110；第 2 阶段第 3 台并行机加工工件 4，其释放时间为 154；整个 HFSP 调度问题的 Makespan 值为 154。

可见，置换 HFSP 是一般 HFSP 的一种特殊情况，其最优调度的 Makespan 值必然不小于一般问题最优调度的 Makespan 值。

显然，基于置换编码的 HFSP 调度问题本质上跟传统 FSP 调度问题和 TSP 问题相同，因此该编码方式下的遗传操作和算法框架可采用跟传统 FSP 调度问题和 TSP 问题相同的设计方案。读者可参阅本书前文有关章节。譬如，采用随机初始化、PMX 或 LOX 等交叉操作、SWAP 或 INV 等变异操作、轮盘赌或锦标赛选择、种

群整体替换等操作；采用传统 GA 和局部搜索算法相结合的框架；也可将传统启发式方法修改后嵌入到 GA 中。

4.7.4 基于复合码的遗传算法设计

本节介绍一种基于工件编号和加工工序的组合的复合码的遗传算法设计。

对于 HFSP 调度问题，每一阶段均要解决工件的排序以及并行机的分配。因此，对于 n 个工件 m 个阶段的 HFSP 问题，可由 m 个子排列联合构成染色体，其中每个子排列为所有工件的一个随机完全排列，并且由 M_j-1 个分隔符“ * ”将其分成 M_j 段，每一子段表示该阶段上某一台并行机上的工件加工顺序。显然，多次重复上述染色体生成过程可产生多个不同的调度方案，进而构成遗传算法的种群。譬如，若考虑一个 6 工件 2 阶段的 HFSP 问题，第 1 和第 2 阶段分别有 2 台和 3 台并行机，则染色体[11 31 51 * 41 61 21 12 52 * 42 32 * 62 22]表示：第 1 阶段的第 1 台机器依次加工工件 1,3,5，第 1 阶段的第 2 台机器依次加工工件 4,6,2，第 2 阶段的第 1 台机器依次加工工件 1,5，第 2 阶段的第 2 台机器依次加工工件 4,3，第 2 阶段的第 3 台机器依次加工工件 6,2。显然，这种编码也可以有效地表示可行调度。

鉴于这种编码的特殊性，交叉变异操作需要作特殊设计。

交叉操作可采用如下方案。首先随机确定交叉工序(即加工阶段)，两后代个体分别继承父代个体非交叉工序上的所有复合基因，对于交叉工序，后代个体首先分别继承两父代个体的分隔符位置，然后所确定的加工阶段上两父代个体的工件排列进行 PMX 或 LOX 或 OX 等交叉操作，从而得到后代个体在该阶段上的所有工件排列，进而用分隔符将它们分隔来进行并行机分配。譬如：若两父代个体为[11 31 51 * 41 61 21 12 52 * 42 32 * 62 22]和[61 51 * 11 41 21 31 42 * 32 12 22 * 52 62]，交叉工序为 2，则由于[1 5 4 3 6 2]与[4 3 1 2 5 6]进行 PMX(随机交叉位置为 2 和 5)的结果为[4 6 1 2 5 3]和[1 2 4 3 6 5]，因而后代个体为[11 31 51 * 41 61 21 42 62 * 12 22 * 52 32]和[61 51 11 41 21 31 12 * 22 42 32 * 62 52]。显然，这种交叉操作保证了后代个体的合法性，并且可以利用常规 PLS 置换码的任意交叉操作。但是，由于父代个体和后代个体中同一并行机上加工的工件数量未变，因此也可以考虑分隔符的其他继承方案来实现同一并行机上加工工件数量的可变性。当然，下面的变异操作一定程度上可以做到这一点，而交叉和变异的结合使用将使得在所有染色体间进行转换的可能得以实现。

变异操作采用如下方案。随机确定变异工序，后代个体首先继承父代个体非变异工序上的所有复合基因，然后对所确定工序上的某些随机位置上的复合基因，即工件号＋工序号＋ * (不是每个复合基因都带 * 号)进行交换或逆序或插入操作。譬如：若父代个体为[11 31 51 * 41 61 21 12 52 * 42 32 * 62 22]，变异工序号为 2，采用 SWAP 变异操作，互换位置为 2 和 3，则后代个体为[11 31 51 * 41 61 21 12 42 52 *

32 * 62 22]。显然，此例子将使第2阶段的第1台机器从依次处理工件1,5转化为依次处理工件1,4,5，而第2台机器由依次处理工件4,3转化为仅处理工件3。因此，该变异操作将改变某些并行机上加工工件的数量及其顺序。

至于其他遗传操作和算法框架，则完全可采取前文介绍的方案，在此不再给予介绍。另外，读者可基于上述各种方案进行数值仿真，在此也不再给出数值结果。

第5章 并行机调度及其遗传算法

并行机调度是实际生产过程中的一类典型调度问题，它研究 n 个工件在 m 台机器上的加工过程，每个工件仅需在某一台机器上加工一次（不加特殊说明，一般约定所有机器的加工性能相同），要求某调度指标最优。显然，所有工件在各机器上的分配问题以及各机器上工件加工顺序，是解决并行机调度问题的两个本质问题。尽管并行机调度问题是混合 Flow Shop 调度问题的一个特例，但因其有代表性，故调度领域还是将其单独归为一类调度问题。本章围绕并行机调度问题，首先介绍最小化最大完成时间指标下的遗传算法设计，其次介绍最小化最大加权推迟时间指标下的遗传算法设计，然后介绍最小化公共交货期下超前拖后指标下的遗传算法设计，最后介绍一类带工艺约束的并行机调度的遗传算法设计。

5.1 最小化最大完成时间的遗传算法

5.1.1 问题描述

Makespan 指标下的并行多机调度问题已被证明为 NP 完全问题。令 $t(j)$ 为工件 j 所需的加工时间，则可定义工件 j 在机器 i 上的广义加工时间为 $x(i,j)t(j)$。其中，若工件 j 在机器 i 上加工，则 $x(i,j)=1$；否则 $x(i,j)=0$。若不考虑同一台机器上加工各工件之间的准备时间或假定准备时间与工件排序无关，则 Makespan 指标跟同一台机器上所加工的工件的排序无关，仅取决于该机器加工工件的数量及其加工时间。

因此，基于 0-1 变量 $x(i,j)$ 的引入，机器 i 上的完工时间为 $\sum_{j=1}^{n}x(i,j)t(j)$。进而，整个加工过程的最大完成时间和相应的调度指标分别为

$$C_{max}=\max_{i=1}^{m}\sum_{j=1}^{n}x(i,j)t(j) \tag{5-1-1}$$

$$\min\max_{i=1}^{m}\sum_{j=1}^{n}x(i,j)t(j) \tag{5-1-2}$$

5.1.2 遗传算法设计

1. 编码和目标值计算

鉴于上述描述，显然问题的解取决于 $x(i,j)$ 的取值。为保证解的合法性，由

$x(i,j)$构成的 $m\times n$ 维矩阵必须要求每列有且仅有一个1,而其余元为0,这决定了相应工件的加工机器号。由于矩阵编码涉及基因过多,且存在大量冗余信息,而且不便于遗传操作的设计,因此刘民等(1998)采用了自然数直接编码来解决这个问题。也即,由 n 个取值为[1,m]之间整数的基因 k_j 构成染色体[k_1 k_2 … k_n],记作 X,每个基因代表该工件加工的机器号。若机器号相同,则表示所对应的工件在同一台机器上加工。

一旦染色体确定,$x(i,j)$的值也就可以确定下来,即

$$x(k_j,j)=1,\quad x(i,j)=0,i\neq k_j,\quad j=1,2,\cdots,m$$

进而,就可以由式(5-1-1)计算该染色体调度方案的 Makespan 性能。

2. 初始种群

初始种群(包括 N 个染色体)一般随机产生。对于每个染色体,它通过在[1,m]间随机选取 n 个允许重复的整数构成。

3. 适配值函数与复制操作

鉴于优化目标为最小化 Makespan 值,因此令适配值函数为

$$f(X)=\alpha\exp(-\beta C_{\max})$$

其中 α,β 为正实数。

复制操作可采用轮盘赌方法,即以概率 $p(X_i)=f(X_i)\Big/\sum f(X_j)$ 选取个体 X_i。具体而言,令 $s(0)=0$,$s(k)=\sum_{i=1}^{k}p(X_i)$,$k=1,2,\cdots,N$,产生均匀分布随机数 $\xi\in[0,1]$;若 $s(k-1)<\xi\leqslant s(k)$,则选取个体 X_k。

4. 交叉与变异操作

鉴于染色体中各基因之间的独立性,交叉操作可采用常用的两点交叉,即首先产生两个[1,n]之间的不同随机整数作为交叉位置,然后交换交叉位置之间的片段,而保留交叉位置之外的片段。

对于变异操作,可对每一基因以一定概率用[1,m]之间区别于该基因的整数加以替代。

5. 终止准则

采用设置最大进化代数 $N_{\max}$。

5.1.3 计算实例

首先考虑7个工件3台机器的小规模问题,加工数据见表5.1.1;其次考虑30个工件10台机器的较大规模问题,加工数据见表5.1.2。

表 5.1.1 7个工件3台机器问题的加工数据

工件号	1	2	3	4	5	6	7
加工时间	6	6	4	4	4	3	3

表 5.1.2 30个工件10台机器问题的加工数据

工件号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
加工时间	3	2	6	4	5	7	9	13	4	12	10	18	22	11	8
工件号	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
加工时间	26	14	6	17	27	11	17	26	16	7	23	15	18	15	13

对于小规模问题，规则性方法 LPT 得到的最好 Makespan 值为 11，而遗传算法很容易得到最优值 10，并且优化速度较快，且对算法参数依赖性不强。

对于较大规模问题，取种群数为 20，最大进化代数为 77，交叉概率为 0.92，变异概率为 0.05，则 15 次随机实验的平均 Makespan 值为 42.3，最优值为 41。可见，GA 的性能仍是比较满意的，但仿真时发现，解的质量对算法参数有较大依赖性。因此，对于大规模问题需要合理选取算法参数，这也是 GA 求解其他优化问题时面临的一个重要问题。

5.2 最小化最大加权推迟时间的遗传算法

5.2.1 问题描述

考虑 n 个工件在 m 台相同并行机上的加工问题($m<n$)。各工件的加工时间和权系数分别为 $p_1,p_2,\cdots,p_n$ 以及 $w_1,w_2,\cdots,w_n$，公共交货期为 d，要求确定一个最优调度，使得加权绝对推迟时间最小。

给定一个调度 $\sigma\in\Pi$，令 c_j 为该调度下工件 j 的完工时间，则该调度问题可描述如下：

$$\min_{\sigma\in\Pi} f(\sigma)=\max\{w_j\mid c_j-d\mid;\quad j=1,2,\cdots,n\} \tag{5-2-1}$$

尽管该调度指标为非正规性能指标，但显然最优调度不允许工件间的加工出现机器空闲的情况。下面，介绍该调度问题的一种遗传算法(Cheng 等，1997)。

5.2.2 遗传算法设计

1. 编码

对照 5.2.1 节研究的并行机调度问题，本节研究的问题不仅要确定所有工件在各机器上的分配，而且要确定同一台机器上各工件的排列顺序，因为调度指标为最小

化加权绝对推迟时间。Cheng 等(1997)为此提出了一种扩展顺序表达方法，即首先将 n 个工件的编号进行全排列，然后随机插入 $m-1$ 个分隔符“ * ”，用以区分不同机器上的加工工件。譬如，染色体[5 8 1 * 3 7 4 6 * 2 9]表示一个 9 个工件 3 台机器的调度，机器 1 上依次加工工件 5,8,1，机器 2 上依次加工工件 3,7,4,6，而机器 3 上依次加工工件 2 和 9。

在算法开始阶段，可用上述方式随机产生一组染色体构成初始种群。

2. 交叉操作

采用如下交叉步骤：

首先，从一个父代个体中继承分割符“ * ”的位置，从而决定后代个体各机器上工件的分配数量。

然后，后代继承上步中父代个体的某个子调度(即某台机器上的工件号及其次序)。

最后，对另一个父代个体，从左到右依次扫描与上步不同的工件号，并依次填入后代个体相应的位置。

图 5.2.1 示意了上述交叉过程。

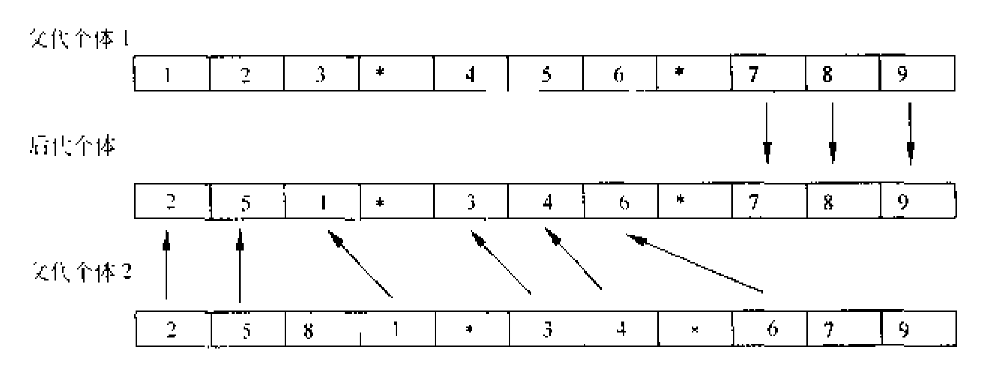


图 5.2.1 交叉操作示意图

3. 变异操作

(1) 交换变异

交换变异即交换两个随机位置上的基因，包括分割符。

- 若交换基因为同一机器上的两个工件，则变异仅改变该机器上的工件加工次序。
- 若所交换的两个工件不在同一台机器上，则变异的结果将导致两台机器加工过程的变化，但各机器上加工工件数量不变。
- 若两交换的基因均为分割符，则变异没有意义，约定 GA 禁止该变异操作。
- 若交换基因为一个工件号和一个分割符，则某些机器上的工件加工数量和顺序均将发生变化。

(2) 启发式变异

启发式变异通过启发式方法采用贪婪的思想来调整工件的顺序，从而形成每台机器上的V型子调度（单机调度问题的最优条件）。也即，公共交货期前的工件按 w_i/p_i 的非增顺序排列，公共交货期后的工件按 w_i/p_i 非减的顺序排列。

令 $d-x$ 表示最早工件的起始时间，$d+y$ 表示调度的最后一个工件的完工时间，则启发式变异的步骤如下。

[**启发式变异**]：

步骤1：按 w_i/p_i 递减顺序排列工件，记调度的工件为 $[J_1, J_2, \cdots, J_n]$，令 $x=0$，$y=0$。

步骤2：把工件 J_1 放在交货期 d 前的位置，令 $x=x+p_1$。

步骤3：i 由2至 n 重复以下步骤：

若 $y+p_i<x$，则把工件 J_i 加在调度列表的最后，并令 $y=y+p_i$；

否则，把该工件加在调度列表的最前面，并令 $x=x+p_i$。

4. 确定交货期

对于单机情况，若工件排列给定，则最优交货期可以确定下来。对于任意两个工件 i 和 j，显然最优交货期满足

$$w_i(d-c_i) = w_j(c_j-d)$$

从而

$$d = (w_i c_i + w_j c_j)/(w_i + w_j)$$

这样，所有工件对的最优交货期就可以确定下来，从而，所有工件的最优交货期可从具有最大绝对推迟时间的交货期中选取。

对于多机问题，首先用上述方法确定各台机器的最佳交货期 d_k，然后定义所有机器的公共交货期为

$$d = \max\{d_k, k=1,2,\cdots,m\}$$

进而，将各机器上工件的最早允许加工时间重新设置为 $d-d_k$。也就是说，许多机器需要推迟起始加工时间，以改善调度性能指标。

5. 适配值函数和复制

基于上述讨论，在工件分配和排列确定好以后，就可以确定公共交货期，进而计算加权绝对推迟时间 $f(\sigma)$。为简单起见，可以令适配值函数为目标值函数的倒数，从而确定个体的适配值。进而，采用轮盘赌方式进行比例复制。在此，不再重复介绍。

5.2.3 计算实例

Cheng 等(1997)采用上述遗传算法求解随机产生的30个工件5台机器的调度问题，遗传种群取50，交叉和变异概率为0.5，最大进化代数为500。

20次随机仿真的结果见表5.2.1。其中,GA表示仅采用交换变异;MA表示两种变异的混合;启发式方法为Li等(1993)的方法。显然,MA方法是最好的。

表5.2.1 仿真统计结果与比较

指　标	MA	GA	启发式方法
最优指标	33.33	43.19	38.5
平均指标	37.26	45.97	
最差指标	38.76	48.12	

5.3 最小化公共交货期下E/T指标的遗传算法

5.3.1 问题描述

自准时制(just in time, JIT)生产在日本获得成功之后,受到国际制造业的普遍重视。JIT生产的指导思想是把库存看作生产过程中的浪费,通过减少库存和准时交货来提高生产效益。JIT生产要求工件尽量准时在其交货期完成,提前和拖后均认为是损失。因此,提前/拖后(E/T)指标下的优化调度是JIT生产领域的重要调度问题,如今受到了理论界和工程界的普遍关注。同样,E/T调度也是并行机调度的重要研究内容。

公共交货期下的E/T并行机调度问题可描述如下:n个工件在m台完全相同的并行机上加工,每个工件只需在任意一台机器上加工一次,各工件都有确定的加工时间和公共交货期,要求确定最优公共交货期和各机器上加工的工件号及其顺序,使得所有工件的提前/拖后成本最小。

记m_j为机器j上的加工工件数,显然,$\sum_{j=1}^{m} m_j = n$,$k_{j,i}$为机器j上第i次加工的工件号,$j=1,2,\cdots,m$,$i=1,\cdots,m_j$,$k_{j,i}$为1至n的整数。令$p(k_{j,i})$为工件$k_{j,i}$的加工时间,d为公共交货期,则工件$k_{j,i}$的完工时间为$\sum_{l=1}^{i} p(k_{j,l})$,其拖后时间和提前时间分别为

$$T[k_{j,i}] = \max\left\{0, \sum_{l=1}^{i} p(k_{j,l}) - d\right\} \geqslant 0 \tag{5-3-1}$$

$$E[k_{j,i}] = \max\left\{0, d - \sum_{l=1}^{i} p(k_{j,l})\right\} \geqslant 0 \tag{5-3-2}$$

进而,E/T指标可描述如下:

$$f(\sigma, d) = \sum_{j=1}^{m} \sum_{i=1}^{m_j} \{\alpha d + \beta E[k_{j,i}] + \gamma T[k_{j,i}]\}$$

$$= n\alpha d + \sum_{j=1}^{m}\sum_{i=1}^{m_j}\{\beta E[k_{j,i}] + \gamma T[k_{j,i}]\} \quad (5\text{-}3\text{-}3)$$

其中，σ 表示一个调度方案；α,β,γ 为非负加权系数，分别用于对交货期设置、提前完工和拖后完工的惩罚。

5.3.2 遗传算法设计

本节介绍刘民等(2000)提出的求解上述问题的一种混合遗传算法。

1. 编码

采用由 3 个基因串构成的复合染色体。其中，每台机器上加工的工件号依次排列作为第 1 个基因串；每台机器上加工的工件总数依次排列作为第 2 个基因串；r 位二进制表示的公共交货期则作为第 3 个基因串。即染色体表示为$[k_{1,1},\cdots,k_{1,m_1},k_{2,1},\cdots,k_{2,m_2},\cdots,k_{m,1},\cdots,k_{m,m_m}|m_1,\cdots,m_m|l_1,l_2,\cdots,l_r]$，其中 $l_i\in\{0,1\}$，r 为码长。可见，整个染色体包括 $n+m+r$ 个基因。

2. 种群初始化

首先考虑染色体中的第 2 个基因串。由于每个工件均要被加工 1 次，也即 $\sum_{j=1}^{m} m_j = n$，所以染色体必须满足该条件。一个简单的方法是：首先，按负荷均衡的思想，取 $m_j = \mathrm{int}[n/m]$，$j=1,2,\cdots,m-1$，令 $m_m = n-(m-1)\mathrm{int}[n/m]$，其中 int[] 表示取整；然后，随机选取 m_i 和 m_j，分别对它们一增一减相同的随机正数，若得到的值小于 1 则不进行此次变化。显然，约束等式仍满足。

第 1 个基因串是所有工件的随机置换排列。当第 2 个基因串确定后，在目标值计算时可根据 m_i 的值，将第 1 基因串中的各工件按先后次序分配给各机器。

第 3 个基因串可以是一个简单的 0-1 随机字符串。

至此，一个染色体就产生了。种群初始化，也即按上述方式随机产生 N 个染色体。由于操作的随机性，初始个体具有一定的多样性。

3. 适配值函数和选择操作

一旦染色体确定，在 α,β,γ 给定的情况下，其目标函数值 f 就可以计算出来。由于遗传算法是对适配值函数的最大化过程，因此可令适配值为 ae^{-bf}，其中 a,b 为正实数。

一旦个体的适配值确定，就可以按前文的轮盘赌方式对种群中的个体进行比例选择。在此不再赘述。

4. 交叉操作

鉴于染色体中 3 个基因串的不同含义，简单的思路就是采用三段独立交叉。由于第 2 基因串涉及等式约束，故为了简单起见，仅对第 1 和第 3 基因串进行交叉，第 2 基因串则完全由后代个体所继承。

由于第1基因串为所有工件的全排列，因此可以对两父代个体的第1基因串执行PMX交叉，当然也可以采用其他置换交叉操作。

由于第3基因串为0-1字符串，因此可采用标准遗传算法的一点交叉方法。

5. 变异操作

变异操作也采用三段变异策略。

对于第1基因串，采用混合变异策略。若产生{1，2}中的随机整数为1，则采用SWAP变异，即对两个随机基因进行互换；否则，采用INVERSE变异，即对子串进行逆序。

对于第2基因串，由于未进行交叉操作，因此其变异操作对全空间搜索很重要。具体实现采用跟初始化相同的思想。首先，随机选取两个不同的基因，分别对其进行一增一减相同的随机正数，来保证等式约束，若所得基因值小于1，则不进行这次变异。

对于第3基因串，采用标准遗传算法的变异方法，即对每一位基因以变异概率p_m进行0 1变换。

6. 补充交叉操作

熟知，对于具体问题，遗传算法虽能够表现出较强的全空间搜索能力，但局部搜索能力较差。因此，在遗传算法的常规交叉和变异操作之后，将所得个体与Cheng(1989)的启发式方法得到的个体再次进行p_{cc}概率下的上述交叉操作，以补充遗传算法的搜索能力。

5.3.3 计算实例

刘民等(2000)采用上述遗传算法设计策略，对几个不同算例在586计算机上进行了仿真研究，并与Cheng(1989)的启发式方法进行了比较。

20次随机仿真的统计结果如表5.3.1所列，其中工件的加工时间均随机产生。

表5.3.1 仿真结果与比较

问题	启发式方法		遗传算法			
工件数×机器数	最优值	公共交货期	最优值	平均值	CPU时间/s	公共交货期
8×3	113	6	106	107.4	4	8
20×6	1218	25	1170	1182	7	28
30×8	1762	22	1697	1703	9	22
40×10	2865	30	2775	2775.8	10	30

仿真结果表明，遗传算法的结果明显优于启发式方法，平均值与最优值之间的差距也不大，而且计算所需的时间也可以接受。因此，上述算法设计是有效的。

5.4 一类带工艺约束的并行机调度的遗传算法

5.4.1 问题描述

前面介绍的并行机调度问题中，机器的加工能力均相同，但在工业生产过程中，特别是化工、钢铁、纺织等行业经常遇到加工能力不同的并行机生产系统，譬如各机器可加工的工件的类型受工艺约束。显然，对于这类问题，前面讨论的遗传算法需要重新设计。

刘民等(2001)研究了如下一类带特殊工艺约束的并行机生产调度问题。考虑 n 个工件在 m 台机器上的加工过程，每个工件总共只有一道工序，可以在任意一台机器上完成加工，各机器的加工能力可以不相同，或者说每台机器可加工的工件受限。记 A 为所有机器的集合，可加工工件 k 的机器集合为 A_k，显然 $A_k \subseteq A$；记 B_j 为机器 j 可加工的工件集合。要求在满足工艺约束的条件下，寻求一个调度方案，即各机器上加工工件的代号及其排序，使得最大完成时间、滞后成本、库存成本的总和最小。

令 m_j 为机器 j 上的加工工件数，$k_{j,i}$ 为机器 j 上第 i 次加工的工件号，t_j 为机器 j 的完工时间，显然 $j=1,2,\cdots,m, i=1,2,\cdots,m_j, k_{j,i} \in B_j$。记 $p(k_{j,i})$ 为工件 $k_{j,i}$ 在机器 j 上的加工时间，则该工件的完工时间为 $\sum_{l=1}^{i} p(k_{j,l})$。设工件 $k_{j,i}$ 的交货期为 $d(k_{j,i})$，最长库存时间为 $t_B(k_{j,i})$，则该工件的拖后完成时间为 $\max\left\{0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i})\right\}$，超库存时间为 $\max\left\{0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i}) - t_B(k_{j,i})\right\}$。于是，在满足工艺约束条件下，调度的目标函数可描述如下：

$$f = \lambda_1 \max_{1 \leq j \leq m} t_j + \sum_{k_{j,i}=1}^{n} \left\{ \lambda_2 \max\left[0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i})\right] \right.$$

$$\left. + \lambda_3 \max\left[0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i}) - t_B(k_{j,i})\right] \right\}$$

令 $\lambda_1=1, \alpha=\lambda_2/\lambda_1, \beta=\lambda_3/\lambda_1$，则上式可改写如下：

$$f = \max_{1 \leq j \leq m} t_j + \sum_{k_{j,i}=1}^{n} \left\{ \alpha \max\left[0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i})\right] \right.$$

$$\left. + \beta \max\left[0, \sum_{l=1}^{i} p(k_{j,l}) - d(k_{j,i}) - t_B(k_{j,i})\right] \right\}$$

下面介绍该并行机调度问题的遗传算法设计。

5.4.2 遗传算法设计

1. 编码

首先把每个工件 k 的加工机器 a_k 和在该机器上相应的加工次序号定义为二维数组$\begin{bmatrix} a_k \\ b_k \end{bmatrix}$,则按 1 至 n 的自然数递增顺序排列的 n 个工件号相应的二维数组将组成一个 $n \times 2$ 维矩阵$\begin{bmatrix} a_1 & a_2 \cdots & a_n \\ b_1 & b_2 \cdots & b_n \end{bmatrix}$,该矩阵可作为遗传算法的染色体,用于表征一个调度方案。其中,由于 A_k 为可加工工件 k 的机器集合,m_{a_k} 为机器 a_k 加工的工件总数,故 $a_k \in A_k$,$b_k \in [1, m_{a_k}]$。

2. 种群初始化

首先,随机产生 N 个 n 维向量$(a_1\ a_2 \cdots\ a_n)$作为每个染色体的第 1 行,其中 a_k 为满足 $a_k \in A_k$ 的任意自然数。其次,由于机器 j 上加工工件总数为 m_j,故每一染色体中第 2 行与机器 j 相对应的元素随机地从$[1, m_j]$中选取不重复的自然数,从而确定相应工件在该机器上的加工次序。显然,这种初始化过程自动满足了加工约束,并且所具有的随机性使得种群具有一定的分散性。

3. 交叉操作

鉴于编码的特殊性,染色体各行的基因串代表了不同的含义,因此需要对交叉操作进行特殊设计。在此对染色体的两行分别进行如下交叉。

由于每个工件可加工的机器集合事先给定,因此可对两个体第 1 行基因串进行两点交叉,这样后代个体仍能满足工艺约束。

对于第 2 行基因串,先采用两点交叉使后代尽可能保留父代信息(指同一台机器上被加工工件的先后次序),然后按以下原则进行修正。假定经过两点交叉后第 2 行基因串中与机器 j 相应的 m_j 个元素为 $b_{j1}, b_{j2}, \cdots, b_{jl}, \cdots, b_{jm_j}$,则令其中最小值对应的新基因值为 1,次小值对应的新基因值为 2,依此类推。若存在 p 个相同值对应新基因值 q,则从$[q, q+p-1]$中随机选取互不相同的 p 个值作为修正后的相应位置的 p 个基因值。

例 假设交叉父代个体为

$$\begin{bmatrix} A_1 \\ B_1 \end{bmatrix} = \left[\begin{array}{cc:ccc:ccc} 1 & 3 & 3 & 1 & 2 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 & 3 & 1 & 3 & 2 \end{array}\right], \quad \begin{bmatrix} A_2 \\ B_2 \end{bmatrix} = \left[\begin{array}{cc:ccc:ccc} 3 & 2 & 1 & 1 & 2 & 3 & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 & 2 & 3 & 1 \end{array}\right]$$

交叉后个体为

$$\begin{bmatrix} A_1' \\ B_1' \end{bmatrix} = \left[\begin{array}{cc:ccc:ccc} 1 & 3 & 1 & 1 & 2 & 2 & 1 & 2 \\ \underline{2} & 1 & \underline{\underline{3}} & \underline{2} & 1 & 1 & \underline{\underline{3}} & 2 \end{array}\right], \quad \begin{bmatrix} A_2' \\ B_2' \end{bmatrix} = \left[\begin{array}{cc:ccc:ccc} 3 & 2 & 3 & 1 & 2 & 3 & 3 & 1 \\ 1 & 2 & 2 & 1 & 3 & 2 & 3 & 1 \end{array}\right]$$

则修正交叉后个体的第 2 行基因串得

$$\begin{bmatrix} A_1' \\ B_1'' \end{bmatrix} = \begin{bmatrix} 1 & 3 & \vdots & 1 & 1 & 2 & \vdots & 2 & 1 & 2 \\ \underline{1} & 1 & \vdots & \underline{\underline{4}} & \underline{2} & 1 & \vdots & 2 & \underline{\underline{3}} & 3 \end{bmatrix}, \qquad \begin{bmatrix} A_2' \\ B_2'' \end{bmatrix} = \begin{bmatrix} 3 & 2 & \vdots & 3 & 1 & 2 & \vdots & 3 & 3 & 1 \\ 1 & 1 & \vdots & 2 & 1 & 2 & \vdots & 3 & 4 & 2 \end{bmatrix}$$

其中带下划线"_"和"="的位置表示因都要在机器 1 上加工而需修正基因的情形。

4. 变异操作

由于个体第 1 行基因串各元素均属于相应的机器集合 $A_k(k=1,2,\cdots,n)$，也即基因取值具有一定的约束性，故简单的变异操作可设计为：对第 1 行的各基因，随机产生(0，1)之间的实数，若该实数小于变异概率 p_m，则随机地从 A_k 中选取基因 a_k' 替代 a_k。

染色体的第 2 行基因串不进行变异操作，但为了满足问题的约束，为了尽可能保留父代信息(指同一台机器上加工工件的先后顺序)，在第 1 行基因串变异完毕后采用交叉操作的修正方案进行修正，即若变异后个体的第 2 行基因串中与机器 j 相应的 m_j 个元素为 $b_{j1},b_{j2},\cdots,b_{jm}$，令 $b_{jl}(1\leqslant l\leqslant m)$的最小值对应的新基因为 1，次小值对应的新基因为 2，依此类推。若存在 p 个相同值对应新基因值 q，则从$[q,q+p-1]$中随机选取互不相同的 p 个值作为修正后的相应位置的 p 个基因值。

例 变异前个体为

$$\begin{bmatrix} A_1 \\ B_1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & \underline{3} & 1 & 2 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 & 3 & 1 & 3 & 2 \end{bmatrix}$$

第 1 行基因串变异后得

$$\begin{bmatrix} A_1' \\ B_1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & \underline{1} & 1 & 2 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 & 3 & 1 & 3 & 2 \end{bmatrix}$$

修正后得

$$\begin{bmatrix} A_1' \\ B_1' \end{bmatrix} = \begin{bmatrix} 1 & 3 & \underline{1} & 1 & 2 & 2 & 1 & 2 \\ 2 & 1 & 3 & 1 & 3 & 1 & 4 & 2 \end{bmatrix}$$

其中带下划线的位置表示发生变异的情况。

5.4.3 计算实例

在钢铁厂的冷轧精整生产中，根据钢卷的宽度和厚度分为一定的类别，不同类别的钢卷有不同的加工条件，并分配给不同的生产线，同时生产调度又受到产品交货期和库存时间的限制。假设某横切机组有 4 条生产线，第 1 条只能加工宽度小于 1.0m 的钢卷，第 2 条只能加工 1.5m 以下的钢卷，第 3 条可加工 2m 以下的，第 4 条可加工所有的钢卷。显然，该问题属于上文讨论的带工艺约束的并行机调度问题，可应用前面设计的遗传算法进行优化。表 5.4.1 为工件加工数据(董斌等，1998)，表 5.4.2 为机器可加工工件的类型，表 5.4.3 为算法优化结果，其中算法 1 为前文设计的遗传算法，算法 2 为董斌等的方法(董斌等，1998)。

表 5.4.1 工件加工数据

钢卷	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
生产时间	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
交货期	2	3	2	0	0	0	3	7	3	9	2	3	2	10	10	10	3	7	3	9
类别	A	A	C	B	C	C	D	D	E	A	A	A	C	B	C	C	D	D	E	A

表 5.4.2 机器可加工工件类别

机 器	工 件 类 别				
M_1	A	B	C	D	E
M_2	A	B	C	D	
M_3	A	B	C		
M_4	A	B			

表 5.4.3 算法优化结果

性能	成本函数平均值	成本函数最差值	拖期工件数平均值	拖期工件数最差值	机器平均利用率(%)
算法 1	5.75	6.50	1.20	2	98
算法 2	6.20	8.00	1.55	4	86

由此可见，在考虑加工工艺约束和产品交货期、库存时间的限制时，前文设计的遗传算法具有很好的优化性能，成本函数值、拖期工件数量和机器利用率均比传统遗传算法要好。

尽管本章仅讨论了部分类型的并行机调度问题的遗传算法设计，但鉴于遗传算法强大的优化性能和算法的通用性，随着遗传算法本身研究的深入和生产调度问题的拓宽，相信遗传算法具有广阔的应用前景，并完全可适用于其他类型的调度问题。

附　　录

附录 1　典型 Job Shop 调度问题

附录 1 给出本书 3.2 节介绍的若干典型 Job Shop 调度问题的加工数据、工艺约束等信息，以便读者进行相关研究。对于每个算例，第 k 行数据（各行数据用分号隔开）表示第 k 号工件的各操作按加工先后顺序对应的机器号（即工艺约束）及相应加工时间，其中机器编号由 0 至 $m-1$。譬如，FT06 算例的第 1 行数据表示工件 1 先在机器 2 上加工 1 单位时间，然后在机器 0 上加工 3 单位时间，然后在机器 1 上加工 6 单位时间，依次类推，最后在机器 4 上加工 6 单位时间。

附录 1.1　FT 类（3 个子问题）

• FT06 或 MT06(6×6)
2 1 0 3 1 6 3 7 5 3 4 6;
1 8 2 5 4 10 5 10 0 10 3 4;
2 5 3 4 5 8 0 9 1 1 4 7;
1 5 0 5 2 5 3 3 4 8 5 9;
2 9 1 3 4 5 5 4 0 3 3 1;
1 3 3 3 5 9 0 10 4 4 2 1;

• FT10 或 MT10(10×10)
0 29 1 78 2 9 3 36 4 49 5 11 6 62 7 56 8 44 9 21;
0 43 2 90 4 75 9 11 3 69 1 28 6 46 5 46 7 72 8 30;
1 91 0 85 3 39 2 74 8 90 5 10 7 12 6 89 9 45 4 33;
1 81 2 95 0 71 4 99 6 9 8 52 7 85 3 98 9 22 5 43;
2 14 0 6 1 22 5 61 3 26 4 69 8 21 7 49 9 72 6 53;
2 84 1 2 5 52 3 95 8 48 9 72 0 47 6 65 4 6 7 25;
1 46 0 37 3 61 2 13 6 32 5 21 9 32 8 89 7 30 4 55;
2 31 0 86 1 46 5 74 4 32 6 88 8 19 9 48 7 36 3 79;
0 76 1 69 3 76 5 51 2 85 9 11 6 40 7 89 4 26 8 74;
1 85 0 13 2 61 6 7 8 64 9 76 5 47 3 52 4 90 7 45;

• FT20 或 MT20(20×5)
0 29 1 9 2 49 3 62 4 44;
0 43 1 75 3 69 2 46 4 72;
1 91 0 39 2 90 4 12 3 45;
1 81 0 71 4 9 2 85 3 22;
2 14 1 22 0 26 3 21 4 72;
2 84 1 52 4 48 0 47 3 6;
1 46 0 61 2 32 3 32 4 30;
2 31 1 46 0 32 3 19 4 36;
0 76 3 76 2 85 1 40 4 26;
1 85 2 61 0 64 3 47 4 90;
1 78 3 36 0 11 4 56 2 21;
2 90 0 11 1 28 3 46 4 30;
0 85 2 74 1 10 3 89 4 33;
2 95 0 99 1 52 3 98 4 43;
0 6 1 61 4 69 2 49 3 53;
1 2 0 95 3 72 4 65 2 25;
0 37 2 13 1 21 3 89 4 55;
0 86 1 74 4 88 2 48 3 79;
1 69 2 51 0 11 3 89 4 74;
0 13 1 7 2 76 3 52 4 45;

附录 1.2　LA 类(40 个子问题)

• LA01(10×5)	• LA02(10×5)
1 21 0 53 4 95 3 55 2 34;	0 20 3 87 1 31 4 76 2 17;
0 21 3 52 4 16 2 26 1 71;	4 25 2 32 0 24 1 18 3 81;
3 39 4 98 1 42 2 31 0 12;	1 72 2 23 4 28 0 58 3 99;
1 77 0 55 4 79 2 66 3 77;	2 86 1 76 4 97 0 45 3 90;
0 83 3 34 2 64 1 19 4 37;	4 27 0 42 3 48 2 17 1 46;
1 54 2 43 4 79 0 92 3 62;	1 67 0 98 4 48 3 27 2 62;
3 69 4 77 1 87 2 87 0 93;	4 28 1 12 3 19 0 80 2 50;
2 38 0 60 1 41 3 24 4 83;	1 63 0 94 2 98 3 50 4 80;
3 17 1 49 4 25 0 44 2 98;	4 14 0 75 2 50 1 41 3 55;
4 77 3 79 2 43 1 75 0 96;	4 72 2 18 1 37 3 79 0 61;

• LA03(10×5)	• LA04(10×5)
1 23 2 45 0 82 4 84 3 38；	0 12 2 94 3 92 4 91 1 7；
2 21 1 29 0 18 4 41 3 50；	1 19 3 11 4 66 2 21 0 87；
2 38 3 54 4 16 0 52 1 52；	1 14 0 75 3 13 4 16 2 20；
4 37 0 54 2 74 1 62 3 57；	2 95 4 66 0 7 3 7 1 77；
4 57 0 81 1 61 3 68 2 30；	1 45 3 6 4 89 0 15 2 34；
4 81 0 79 1 89 2 89 3 11；	3 77 2 20 0 76 4 88 1 53；
3 33 2 20 0 91 4 20 1 66；	2 74 1 88 0 52 3 27 4 9；
4 24 1 84 0 32 2 55 3 8；	1 88 3 69 0 62 4 98 2 52；
4 56 0 7 3 54 2 64 1 39；	2 61 4 9 0 62 1 52 3 90；
4 40 1 83 0 19 2 8 3 7；	2 54 4 5 3 59 1 15 0 88；

• LA05(10×5)
1 72 0 87 4 95 2 66 3 60；
4 5 3 35 0 48 2 39 1 54；
1 46 3 20 2 21 0 97 4 55；
0 59 3 19 4 46 1 34 2 37；
4 23 2 73 3 25 1 24 0 28；
3 28 0 45 4 5 1 78 2 83；
0 53 3 71 1 37 4 29 2 12；
4 12 2 87 3 33 1 55 0 38；
2 49 3 83 1 40 0 48 4 7；
2 65 3 17 0 90 4 27 1 23；

• LA06(15×5)	• LA07(15×5)
1 21 2 34 4 95 0 53 3 55；	0 47 4 57 1 71 3 96 2 14；
3 52 4 16 1 71 2 26 0 21；	0 75 1 60 4 22 3 79 2 65；
2 31 0 12 1 42 3 39 4 98；	3 32 0 33 2 69 1 31 4 58；
3 77 1 77 4 79 0 55 2 66；	0 44 1 34 4 51 3 58 2 47；
4 37 3 34 2 64 1 19 0 83；	3 29 1 44 0 62 2 17 4 8；
2 43 1 54 0 92 3 62 4 79；	1 15 2 40 0 97 4 38 3 66；
0 93 3 69 1 87 4 77 2 87；	2 58 1 39 0 57 4 20 3 50；
0 60 1 41 2 38 4 83 3 24；	2 57 3 32 4 87 0 63 1 21；
2 98 3 17 4 25 0 44 1 49；	4 56 0 84 2 90 1 85 3 61；
0 96 4 77 3 79 1 75 2 43；	4 15 0 20 1 67 3 30 2 70；
4 28 2 35 0 95 3 76 1 7；	4 84 0 82 1 23 2 45 3 38；
0 61 4 10 2 95 1 9 3 35；	3 50 2 21 0 18 4 41 1 29；
4 59 3 16 1 91 2 59 0 46；	4 16 1 52 0 52 2 38 3 54；
4 43 1 52 0 28 2 27 3 50；	4 37 0 54 3 57 2 74 1 62；
0 87 1 45 2 39 4 9 3 41；	4 57 1 61 0 81 2 30 3 68；

• LA08(15×5)	• LA09(15×5)
3 92 2 94 0 12 4 91 1 7;	1 66 3 85 2 84 0 62 4 19;
2 21 1 19 0 87 3 11 4 66;	3 59 1 64 2 46 4 15 0 25;
1 14 3 13 0 75 4 16 2 20;	4 88 3 80 1 73 2 53 0 41;
2 95 4 66 0 7 1 77 3 7;	0 14 1 67 2 57 3 74 4 47;
2 34 4 89 3 6 1 45 0 15;	0 84 4 64 2 41 3 84 1 78;
4 88 3 77 2 20 1 53 0 76;	0 63 3 28 1 46 2 26 4 52;
4 9 3 27 0 52 1 88 2 74;	3 10 2 17 4 73 1 11 0 64;
3 69 2 52 0 62 1 88 4 98;	2 67 1 97 3 95 4 38 0 85;
3 90 0 62 4 9 2 61 1 52;	2 95 4 46 0 59 1 65 3 93;
4 5 2 54 3 59 0 88 1 15;	2 13 4 85 3 32 1 85 0 60;
0 41 1 50 4 78 3 53 2 23;	4 49 3 41 2 61 0 66 1 90;
0 38 4 72 2 91 3 68 1 71;	1 17 0 23 3 70 1 99 2 49;
0 45 3 95 4 52 2 25 1 6;	4 40 3 73 0 73 1 98 2 68;
3 30 1 66 0 23 4 36 2 17;	3 57 1 9 2 7 0 13 4 98;
2 95 0 71 3 76 1 8 4 88;	0 37 1 85 2 17 4 79 3 41;

• LA10(15×5)
1 58 2 44 3 5 0 9 4 58;
1 89 0 97 4 96 3 77 2 84;
0 77 1 87 2 81 4 39 3 85;
3 57 1 21 2 31 0 15 4 73;
2 48 0 40 1 49 3 70 4 71;
3 34 4 82 2 80 0 10 1 22;
1 91 4 75 0 55 2 17 3 7;
2 62 3 47 1 72 4 35 0 11;
0 64 3 75 4 50 1 90 2 94;
2 67 4 20 3 15 0 12 1 71;
0 52 4 93 3 68 2 29 1 57;
2 70 0 58 1 93 4 7 3 77;
3 27 2 82 1 63 4 6 0 95;
1 87 2 56 4 36 0 26 3 18;
3 76 2 36 0 36 4 15 1 8;

• LA11(20×5)	• LA12(20×5)
2 34 1 21 0 53 3 55 4 95;	1 23 0 82 4 84 2 45 3 38;
0 21 3 52 1 71 4 16 2 26;	3 50 4 41 1 29 0 18 2 21;
0 12 1 42 2 31 4 98 3 39;	4 16 3 54 1 52 2 38 0 52;
2 66 3 77 4 79 0 55 1 77;	1 62 3 57 4 37 2 74 0 54;
0 83 4 37 3 34 1 19 2 64;	3 68 1 61 2 30 0 81 4 57;
4 79 2 43 0 92 3 62 1 54;	1 89 2 89 3 11 0 79 4 81;
0 93 4 77 2 87 1 87 3 69;	1 66 0 91 3 33 4 20 2 20;
4 83 3 24 1 41 2 38 0 60;	3 8 4 24 2 55 0 32 1 84;
4 25 1 49 0 44 2 98 3 17;	0 7 2 64 1 39 4 56 3 54;
0 96 1 75 2 43 4 77 3 79;	0 19 4 40 3 7 2 8 1 83;
0 95 3 76 1 7 4 28 2 35;	0 63 2 64 3 91 4 40 1 6;
4 10 2 95 0 61 1 9 3 35;	1 42 3 61 4 15 2 98 0 74;
1 91 2 59 4 59 0 46 3 16;	1 80 0 26 3 75 4 6 2 87;
2 27 1 52 4 43 0 28 3 50;	2 39 4 22 0 75 3 24 1 44;
4 9 0 87 3 41 2 39 1 45;	1 15 3 79 4 8 0 12 2 20;
1 54 0 20 4 43 3 14 2 71;	3 26 2 43 0 80 4 22 1 61;
4 33 1 28 3 26 0 78 2 37;	2 62 1 36 0 63 3 96 4 40;
1 89 0 33 2 8 3 66 4 42;	1 33 3 18 0 22 4 5 2 10;
4 84 0 69 2 94 1 74 3 27;	2 64 4 64 0 89 1 96 3 95;
4 81 2 45 1 78 3 69 0 96;	2 18 4 23 3 15 1 38 0 8;

• LA13(20×5)	• LA14(20×5)
3 60 0 87 1 72 4 95 2 66;	3 5 4 58 2 44 0 9 1 58;
1 54 0 48 2 39 3 35 4 5;	1 89 4 96 0 97 2 84 3 77;
3 20 1 46 0 97 2 21 4 55;	2 81 3 85 1 87 4 39 0 77;
2 37 0 59 3 19 1 34 4 46;	0 15 3 57 4 73 1 21 2 31;
2 73 3 25 1 24 0 28 4 23;	2 48 4 71 3 70 0 40 1 49;
1 78 3 28 2 83 0 45 4 5;	0 10 4 82 3 34 2 80 1 22;
3 71 1 37 2 12 4 29 0 53;	2 17 0 55 1 91 4 75 3 7;
4 12 3 33 1 55 2 87 0 38;	3 47 2 62 1 72 4 35 0 11;
0 48 1 40 2 49 3 83 4 7;	1 90 2 94 4 50 0 64 3 75;
0 90 4 27 2 65 3 17 1 23;	3 15 2 67 0 12 4 20 1 71;
0 62 3 85 1 66 2 84 4 19;	4 93 2 29 0 52 1 57 3 68;
3 59 2 46 4 13 1 64 0 25;	3 77 1 93 0 58 2 70 4 7;
2 53 1 73 3 80 4 88 0 41;	1 63 3 27 0 95 4 6 2 82;
2 57 4 47 0 14 1 67 3 74;	4 36 0 26 3 48 2 56 1 87;
2 41 4 64 3 84 1 78 0 84;	2 36 1 8 4 15 3 76 0 36;
4 52 3 28 2 26 0 63 1 46;	4 78 1 84 3 41 0 30 2 76;
1 11 0 64 3 10 4 73 2 17;	1 78 0 75 4 88 3 13 2 81;
4 38 3 95 0 85 1 97 2 67;	0 54 1 40 2 13 1 82 3 29;
3 93 1 65 2 95 0 59 4 46;	1 26 4 82 0 52 3 6 2 6;
0 60 1 85 2 43 4 85 3 32;	3 54 1 64 0 54 2 32 4 88;

• LA15(20×5)

0 6 2 40 1 81 3 37 4 19;
2 40 3 32 0 55 4 81 1 9;
1 46 4 65 2 70 3 55 0 77;
2 21 4 65 0 64 3 25 1 15;
2 85 0 40 1 44 3 24 4 37;
0 89 4 29 1 83 3 31 2 84;
4 59 3 38 1 80 2 30 0 8;
0 80 2 56 1 77 4 41 3 97;
4 56 0 91 3 50 2 71 1 17;
1 40 0 88 4 59 2 7 3 80;
0 45 1 29 2 8 4 77 3 58;
2 36 0 54 3 96 1 9 4 10;
0 28 2 73 1 98 3 92 4 87;
0 70 3 86 2 27 1 99 4 96;
1 95 0 59 4 56 3 85 2 41;
1 81 2 92 4 32 0 52 3 39;
1 7 4 22 2 12 0 88 3 60;
3 45 0 93 2 69 4 49 1 27;
0 21 1 84 2 61 3 68 4 26;
1 82 2 33 4 71 0 99 3 44;

• LA16(10×10)

1 21 6 71 9 16 8 52 7 26 2 34 0 53 4 21 3 55 5 95;
4 55 2 31 5 98 9 79 0 12 7 66 1 42 8 77 6 77 3 39;
3 34 2 64 8 62 1 19 4 92 9 79 7 43 6 54 0 83 5 37;
1 87 3 69 2 87 7 38 8 24 9 83 6 41 0 93 5 77 4 60;
2 98 0 44 5 25 6 75 7 43 1 49 4 96 9 77 3 17 8 79;
2 35 3 76 5 28 9 10 4 61 6 9 0 95 8 35 1 7 7 95;
3 16 2 59 0 46 1 91 9 43 8 50 6 52 5 59 4 28 7 27;
1 45 0 87 3 41 4 20 6 54 9 43 8 14 5 9 2 39 7 71;
4 33 2 37 8 66 5 33 3 26 7 8 1 28 6 89 9 42 0 78;
8 69 9 81 2 94 4 96 3 27 0 69 7 45 6 78 1 74 5 84;

• LA17(10×10)

4 18 7 21 9 41 2 45 3 38 8 50 5 84 6 29 1 23 0 82;
8 57 5 16 1 52 7 74 2 38 3 54 6 62 9 37 4 54 0 52;
2 30 4 79 3 68 1 61 8 11 6 89 7 89 0 81 9 81 5 57;
0 91 8 8 3 33 7 55 5 20 2 20 4 32 6 84 1 66 9 24;
9 40 0 7 4 19 8 7 6 83 2 64 5 56 3 54 7 8 1 39;
3 91 2 64 5 40 0 63 7 98 4 74 8 61 1 6 6 42 9 15;
1 80 7 39 8 24 3 75 4 75 5 6 6 44 0 26 2 87 9 22;
1 15 7 13 2 20 0 12 8 26 6 61 3 79 9 22 5 8 4 80;
2 62 3 96 4 22 9 5 0 63 6 33 7 10 8 18 1 36 5 40;
1 96 0 89 5 64 3 95 9 23 7 18 8 15 2 64 6 38 4 8;

• LA18(10×10)
6 54 0 87 4 48 3 60 7 39 8 35 1 72 5 95 2 66 9 5;
3 20 9 46 6 34 5 55 0 97 8 19 4 59 2 21 7 37 1 46;
4 45 1 24 8 28 0 28 7 83 6 78 5 23 3 25 9 5 2 73;
9 12 1 37 4 38 3 71 8 33 2 12 6 55 0 53 7 87 5 29;
3 83 2 49 6 23 9 27 7 65 0 48 4 90 5 7 1 40 8 17;
1 66 4 25 0 62 2 84 9 13 6 64 7 46 8 39 5 19 3 85;
1 73 3 80 0 41 2 53 9 47 7 57 8 74 4 14 6 67 5 88;
5 64 3 84 6 16 1 78 0 84 7 26 8 28 9 52 2 41 4 63;
1 11 0 64 7 67 4 85 3 10 5 73 9 38 8 95 6 97 2 17;
4 60 8 32 2 95 3 93 1 65 6 85 7 43 9 85 5 46 0 59;

• LA19(10×10)
2 44 3 5 5 58 4 97 0 9 7 84 8 77 9 96 1 58 6 89;
4 15 7 31 1 87 8 57 0 77 3 85 2 81 5 39 9 73 6 21;
9 82 6 22 4 10 3 70 1 49 0 40 8 34 2 48 7 80 5 71;
1 91 2 17 7 62 5 75 8 17 4 17 3 7 6 72 9 35 0 55;
6 71 1 90 3 75 0 64 2 94 8 15 4 12 7 67 9 20 5 50;
7 70 5 93 8 77 2 29 4 58 6 93 3 68 1 57 9 7 0 52;
6 87 1 63 4 26 5 6 2 82 3 27 7 56 8 48 9 36 0 95;
0 36 5 15 8 41 9 78 3 76 6 84 4 30 7 76 2 36 1 8;
5 88 2 81 3 13 6 82 4 54 7 13 8 29 9 40 1 78 0 75;
9 88 4 54 6 64 7 32 0 52 2 6 8 54 5 82 3 6 1 26;

• LA20(10×10)
6 9 1 81 4 55 2 40 8 32 3 37 0 6 5 19 9 81 7 40;
7 21 2 70 9 63 4 64 1 46 5 65 8 25 0 77 3 55 6 15;
2 85 5 37 0 40 3 24 1 44 6 83 4 89 8 31 7 84 9 29;
4 80 6 77 7 56 0 8 2 30 5 59 3 38 1 80 9 41 8 97;
0 91 6 40 4 88 1 17 2 71 3 50 9 59 8 80 5 36 7 7;
2 8 6 9 3 38 5 77 1 29 8 96 0 45 9 10 4 54 7 36;
4 70 3 92 1 98 5 87 6 99 7 27 8 86 9 96 0 28 2 73;
1 95 7 92 3 85 4 52 6 81 9 32 8 39 0 59 2 41 5 56;
3 60 8 45 0 88 2 12 1 7 5 22 4 93 9 49 7 69 6 27;
0 21 2 61 3 68 5 26 6 82 9 71 8 44 4 99 7 33 1 84;

• LA21(15×10)
2 34 3 55 5 95 9 16 4 21 6 71 0 53 8 52 1 21 7 26;
3 39 2 31 0 12 1 42 9 79 8 77 6 77 5 98 4 55 7 66;
1 19 0 83 3 34 4 92 6 54 9 79 8 62 5 37 2 64 7 43;
4 60 2 87 8 24 5 77 3 69 7 38 1 87 6 41 9 83 0 93;
8 79 9 77 2 98 4 96 3 17 0 44 7 43 6 75 [illegible] 19 5 25;
8 35 7 95 6 9 9 10 2 35 1 7 5 28 4 61 0 95 3 76;
4 28 5 59 3 16 9 43 0 46 8 50 6 52 7 27 2 39 1 91;
5 9 4 20 2 39 6 54 1 45 7 71 0 87 3 41 9 43 8 14;
1 28 5 33 0 78 3 26 2 37 7 8 8 66 6 89 9 42 4 33;
2 94 5 84 6 78 9 81 1 74 3 27 8 69 0 69 7 45 4 96;
1 31 4 24 0 20 2 17 9 25 8 81 5 76 3 87 7 32 6 18;

• LA22(15×10)
9 66 5 91 4 87 2 94 7 21 3 92 1 7 0 12 8 11 6 19;
3 13 2 20 4 7 1 14 9 66 0 75 6 77 5 16 7 95 8 7;
8 77 7 20 2 34 0 15 9 88 5 89 6 53 3 6 1 15 4 76;
3 27 2 74 6 88 4 62 7 52 8 69 5 9 9 98 0 52 1 88;
4 88 6 15 1 52 2 61 7 54 0 62 8 59 5 9 3 90 9 5;
6 71 0 41 4 38 3 53 7 91 8 68 1 50 5 78 2 23 9 72;
3 95 9 36 6 66 5 52 0 45 8 30 4 23 2 25 7 17 1 6;
4 65 1 8 8 85 0 71 7 65 6 28 5 88 3 75 9 27 2 95;
9 37 1 37 4 28 3 51 8 86 2 9 6 55 0 73 7 51 5 90;
3 39 2 15 6 83 9 14 7 53 0 16 4 46 5 24 1 25 8 82;
1 72 4 48 0 87 2 66 9 5 6 54 7 39 8 35 5 95 3 60;

• LA21(15×10)	• LA22(15×10)
5 28 9 97 0 58 4 45 6 76 3 99 2 23 1 72 8 90 7 86;	1 46 3 20 0 97 2 21 9 46 7 37 8 19 4 59 6 34 5 55;
5 27 9 48 8 27 7 62 4 98 6 67 3 48 0 42 1 46 2 17;	5 23 3 25 6 78 1 24 0 28 7 83 8 28 9 5 2 73 4 45;
1 12 8 50 0 80 2 50 9 80 3 19 5 28 6 63 4 94 7 98;	1 37 0 53 7 87 4 38 3 71 5 29 9 12 8 33 6 55 2 12;
4 61 3 55 6 37 5 14 2 50 8 79 1 41 9 72 7 18 0 75;	4 90 8 17 2 49 3 83 1 40 6 23 7 65 9 27 5 7 0 48;

• LA23(15×10)	• LA24(15×10)
7 84 5 58 8 77 2 44 4 97 6 89 3 5 1 58 9 96 0 9;	7 8 9 75 0 72 6 71 4 30 8 43 2 38 5 98 1 26 3 19;
6 21 1 87 4 15 5 39 2 81 3 85 7 31 8 57 9 73 0 77;	6 19 8 73 3 43 0 23 1 85 4 39 5 13 9 26 2 67 7 9;
0 40 5 71 8 34 9 82 3 70 6 22 4 10 7 80 2 48 1 49;	1 50 3 93 5 80 4 7 0 55 2 61 6 57 8 72 9 42 7 46;
5 75 2 17 3 7 6 72 4 11 7 62 8 47 9 35 1 91 0 55;	1 68 7 43 4 99 6 60 5 68 0 91 8 11 3 96 9 11 2 72;
9 20 4 12 6 71 7 67 0 64 2 94 8 15 5 50 3 75 1 90;	7 84 2 34 8 40 5 7 1 70 6 74 3 12 0 43 9 69 4 30;
6 93 5 93 1 57 7 70 8 77 4 58 0 52 2 29 9 7 3 68;	8 60 0 49 4 59 5 72 9 63 1 69 7 99 6 45 3 27 2 9;
7 56 0 95 8 48 4 26 2 82 1 63 9 36 3 27 6 87 5 6;	6 71 2 91 8 65 1 90 9 98 4 8 7 50 0 75 5 37 3 17;
3 76 5 15 9 78 1 8 8 41 2 36 4 30 6 84 0 36 7 76;	8 62 7 90 5 98 3 31 2 91 4 38 9 72 1 9 0 72 6 49;
0 75 7 13 2 81 8 29 4 54 6 82 5 88 1 78 9 40 3 13;	4 35 0 39 9 74 5 25 7 47 3 52 2 63 8 21 6 35 1 80;
2 6 1 26 7 32 6 64 4 54 0 52 5 82 3 6 9 88 8 54;	9 58 0 5 3 50 8 32 1 88 6 20 2 68 5 24 4 53 7 57;
8 62 2 67 5 32 0 62 7 69 3 61 1 35 4 72 9 5 6 93;	7 99 3 91 4 33 5 19 2 18 6 38 0 24 9 35 1 49 8 9;
2 78 9 90 0 85 1 72 8 64 6 63 3 11 7 82 5 88 4 7;	0 68 3 60 2 77 7 10 8 60 5 15 9 72 1 18 6 90 4 18;
4 28 9 11 7 50 6 88 0 44 5 31 2 27 1 66 8 49 3 35;	9 79 1 60 3 56 6 91 2 40 8 86 7 72 0 80 5 89 4 51;
2 14 5 39 6 56 4 62 3 97 9 66 7 69 1 7 8 47 0 76;	4 10 2 92 5 23 6 46 8 40 7 72 3 6 1 23 0 95 9 34;
1 18 8 93 7 58 6 47 3 69 9 57 2 41 5 53 4 79 0 64;	2 24 5 29 9 49 8 55 0 47 6 77 3 77 7 8 1 28 4 48;

• LA25(15×10)
8 14 4 75 3 12 2 38 0 76 5 97 9 12 1 29 7 44 6 66;
5 38 3 82 2 85 4 58 6 87 9 89 0 43 1 80 7 69 8 92;
9 5 1 84 0 43 6 48 4 8 7 7 3 41 5 61 8 66 2 14;
2 42 1 8 0 96 5 19 4 59 7 97 9 73 8 43 3 74 6 41;
6 55 2 70 3 75 8 42 4 37 7 23 1 48 5 5 9 38 0 7;
8 9 2 72 7 31 0 79 5 73 3 95 4 25 6 43 9 60 1 56;
0 97 2 64 3 78 5 21 4 94 9 31 8 53 6 16 7 86 1 7;
3 86 7 85 9 63 0 61 2 65 4 30 5 32 1 33 8 44 6 59;
2 44 3 16 4 11 6 45 1 30 9 84 8 93 0 60 5 61 7 90;
7 36 8 31 4 47 6 52 0 32 5 11 2 28 9 35 3 20 1 49;
8 20 6 49 7 74 4 10 5 17 3 34 0 85 2 77 9 68 1 84;
1 85 5 7 8 71 6 59 4 76 0 17 3 29 2 17 7 48 9 13;
2 15 6 87 7 11 1 39 4 39 8 43 0 19 3 32 9 16 5 64;
6 32 2 92 5 33 8 82 1 83 7 57 9 99 4 91 3 99 0 8;
4 88 7 7 8 27 1 38 3 91 2 69 6 21 9 62 5 39 0 48;

• LA26(20×10)	• LA27(20×10)
8 52 7 26 6 71 9 16 2 34 1 21 5 95 4 21 0 53 3 55;	3 60 4 48 5 95 0 87 1 72 9 5 8 35 7 39 6 54 2 66;
4 55 5 98 3 39 9 79 0 12 8 77 6 77 7 66 2 31 1 42;	7 37 6 34 0 97 5 55 2 21 3 20 4 59 9 46 8 19 1 46;
5 37 4 92 2 64 6 54 1 19 7 43 0 83 3 34 9 79 8 62;	4 45 2 73 1 24 8 28 0 28 3 25 5 23 7 83 9 5 6 78;
1 87 5 77 0 93 3 69 2 87 7 38 8 24 6 41 9 83 4 60;	0 53 2 12 9 12 1 37 8 33 3 71 6 55 5 29 7 87 4 38;
2 98 5 25 6 75 9 77 1 49 3 17 8 79 0 44 7 43 4 96;	4 90 2 49 9 27 7 65 5 7 6 23 0 48 3 83 8 17 1 40;
1 7 4 61 0 95 2 35 9 10 8 35 5 28 3 76 7 95 6 9;	3 85 4 25 2 84 6 64 9 13 1 66 7 46 8 59 0 62 5 19;
5 59 9 43 0 46 4 28 6 52 3 16 2 59 1 91 8 50 7 27;	5 88 6 67 4 14 0 41 1 73 7 57 2 53 3 80 9 47 8 74;
5 9 9 43 8 14 7 71 4 20 6 54 3 41 0 87 1 45 2 39;	1 78 5 64 4 63 6 46 3 84 0 84 8 28 9 52 7 26 2 41;
1 28 8 66 0 78 2 37 9 42 3 26 5 33 6 89 4 33 7 8;	1 11 0 64 6 97 9 38 2 17 4 85 5 73 3 10 8 95 7 67;
4 96 3 27 6 78 5 84 2 94 8 69 1 74 9 81 7 45 0 69;	3 93 2 95 7 43 1 65 8 32 0 59 6 85 5 46 9 85 4 60;
4 24 7 32 9 25 2 17 3 37 8 81 5 76 6 18 1 31 0 20;	2 61 3 41 5 49 4 23 0 66 7 49 8 70 9 99 1 90 6 17;
8 90 5 28 1 72 7 86 2 23 3 99 6 76 9 97 4 45 0 58;	4 13 7 7 1 98 8 57 0 73 3 73 2 68 5 40 9 98 6 9;
2 17 4 98 3 48 1 46 8 27 6 67 7 62 0 42 9 48 5 27;	9 86 6 76 4 14 3 41 1 85 0 37 8 19 2 17 7 54 5 79;
0 80 8 50 3 19 7 98 5 28 2 50 4 94 6 63 1 12 9 80;	1 40 2 53 7 97 5 87 8 96 4 84 3 16 6 66 9 52 0 95;
9 72 0 75 4 61 8 79 6 37 2 50 5 14 3 55 7 18 1 41;	6 33 1 33 3 87 0 18 2 55 8 13 4 77 7 60 9 42 5 74;
3 96 2 14 5 57 0 47 7 65 4 75 8 79 1 71 6 60 9 22;	7 92 5 91 8 79 2 54 4 69 6 79 3 33 1 61 9 39 0 16;
1 31 7 47 8 58 3 32 4 41 5 58 6 34 0 33 2 69 9 51;	6 82 1 41 4 28 5 64 2 78 3 76 7 6 8 49 9 47 0 58;
1 44 7 40 2 17 0 62 8 66 6 15 3 29 9 38 5 8 4 97;	0 52 5 42 8 24 9 91 3 47 6 88 4 91 7 52 2 28 1 35;
2 58 3 50 4 63 9 87 0 57 6 21 7 57 8 32 1 39 5 20;	5 82 2 76 3 86 6 93 4 84 7 38 8 95 9 37 1 21 0 23;
1 85 0 84 5 56 3 61 9 15 7 70 8 30 2 90 6 67 4 20;	9 77 4 8 6 42 7 64 0 70 2 45 8 45 5 28 3 67 1 86;

• LA28(20×10)	• LA29(20×10)
8 32 1 81 4 55 7 40 0 6 5 19 9 81 3 37 2 40 6 9;	8 14 2 38 7 44 0 76 5 97 3 12 4 75 6 66 9 12 1 29;
2 70 3 55 7 21 4 64 1 46 8 25 9 65 0 77 5 65 6 15;	0 43 2 85 3 82 5 38 4 58 9 89 8 92 6 87 7 69 1 80;
7 84 4 89 3 24 1 44 2 85 8 31 9 29 6 83 5 37 0 40;	3 41 7 7 9 5 0 43 2 14 4 8 5 61 1 84 8 66 6 48;
4 80 5 59 0 8 2 30 6 77 3 38 1 80 7 56 9 41 8 97;	2 42 3 74 4 59 6 41 1 8 9 73 8 43 0 96 5 19 7 97;
6 40 2 71 0 91 7 7 9 59 8 80 3 50 5 56 1 17 4 88;	7 23 8 42 4 37 6 55 0 7 5 5 2 70 9 38 3 75 1 48;
7 36 9 10 0 45 6 9 4 54 8 96 2 8 5 77 1 29 3 58;	8 9 6 43 7 31 4 25 5 73 3 95 0 79 2 72 9 60 1 56;
6 99 8 86 3 92 0 28 1 98 4 70 5 87 9 96 2 73 7 27;	1 7 5 21 8 53 6 16 4 94 0 97 3 78 2 64 7 86 9 31;
1 95 3 85 5 56 4 52 0 59 2 41 6 81 8 39 9 32 7 92;	2 65 6 59 7 85 1 33 4 30 8 44 0 61 3 86 9 63 5 32;
1 7 7 69 4 93 6 27 5 22 0 88 8 45 3 60 9 49 2 12;	6 45 2 44 5 61 8 93 1 30 7 90 9 84 4 11 3 16 0 60;
7 33 2 61 8 44 5 26 1 84 6 82 3 68 0 21 9 71 4 99;	4 47 7 36 8 31 1 49 3 20 2 28 6 52 9 35 5 11 0 32;
8 43 0 72 4 30 5 98 9 75 1 26 7 8 6 74 3 19 2 38;	2 77 4 10 9 68 5 17 0 85 1 84 8 20 6 49 7 74 3 34;
6 19 2 67 8 73 1 85 9 26 4 39 7 9 0 23 5 13 3 43;	0 17 5 7 1 85 3 29 2 17 4 76 6 59 8 71 9 13 7 48;
8 72 7 46 5 80 3 93 2 61 4 7 9 42 1 50 0 55 6 57;	6 87 4 39 8 43 7 11 2 15 3 32 5 64 0 19 1 39 9 16;
4 99 0 91 9 11 5 68 7 43 3 96 2 72 8 11 6 60 1 68;	5 33 3 99 6 32 4 91 8 82 2 92 9 99 7 57 1 83 0 8;
9 69 0 43 3 12 8 40 1 70 6 74 2 34 5 7 4 30 7 84;	3 91 5 39 2 69 8 27 7 7 6 21 1 38 9 62 4 88 0 48;
7 99 3 27 4 59 5 72 2 9 6 45 0 49 9 63 1 69 8 60;	2 67 7 80 3 24 0 88 4 18 1 44 8 45 9 64 5 80 6 38;
0 75 3 17 2 91 7 50 8 65 5 37 9 98 1 90 6 71 4 8;	9 59 3 72 6 47 4 40 7 21 5 43 0 51 8 52 1 24 2 15;
9 72 1 9 3 31 6 49 2 91 8 62 7 90 0 72 5 98 4 38;	3 70 2 31 6 20 8 76 1 40 7 43 0 32 5 88 9 5 4 77;
4 35 2 63 5 25 6 35 8 21 7 47 3 52 1 80 0 39 9 74;	4 47 5 64 9 85 3 49 7 58 1 26 0 32 8 80 2 14 6 94;
2 68 5 24 9 58 8 52 0 5 6 20 3 50 7 57 1 88 4 53;	5 59 2 96 0 5 7 79 8 34 4 75 3 26 6 9 9 23 1 11;

• LA30(20×10)

6 32 3 16 1 33 8 12 7 70 4 10 9 75 0 82 5 88 2 20;
8 39 4 81 3 91 5 56 9 69 1 45 6 59 0 86 2 36 7 68;
3 84 2 57 7 41 5 73 4 81 0 88 8 38 9 17 6 83 1 5;
4 20 5 6 2 15 8 19 1 30 0 94 6 45 7 17 3 18 9 88;
9 24 6 49 5 16 4 11 3 60 7 5 8 63 1 25 2 15 0 45;
1 86 8 50 2 77 6 54 9 48 0 93 3 32 7 92 5 45 4 71;
5 86 6 90 3 78 9 88 2 57 0 32 7 57 8 86 4 71 1 39;
2 59 3 18 9 31 4 41 7 20 5 83 8 65 0 54 6 94 1 69;
3 47 4 79 6 76 0 59 1 72 2 8 9 30 5 73 7 57 8 84;
0 59 2 89 4 10 7 45 3 8 5 54 6 88 8 20 9 7 1 62;
5 63 6 9 4 77 3 37 2 5 8 13 9 79 1 24 7 10 0 82;
0 74 1 32 2 61 7 53 4 92 9 20 8 10 3 5 6 45 5 23;
2 85 9 51 0 61 5 99 4 37 6 94 1 98 8 65 3 33 7 75;
0 51 3 24 5 8 6 30 7 12 8 23 2 7 4 17 9 35 1 81;
1 71 5 42 8 68 2 31 6 29 3 63 4 65 9 70 7 27 0 93;
1 28 5 38 4 51 7 70 2 33 8 78 9 45 3 90 6 54 0 72;
0 18 2 90 4 25 6 92 8 85 5 35 7 29 1 81 9 80 3 59;
5 67 2 96 1 38 4 86 0 97 3 94 7 86 6 35 9 82 8 45;
2 92 8 51 4 59 6 52 5 8 9 70 1 75 3 54 7 60 0 33;
3 98 7 80 5 78 0 82 2 7 9 89 1 69 4 51 8 79 6 62;

• LA31(30×10)

4 21 7 26 9 16 2 34 3 55 8 52 5 95 6 71 1 21 0 53;
8 77 5 98 1 42 7 66 2 31 3 39 6 77 9 79 4 55 0 12;
2 64 4 92 3 34 1 19 8 62 6 54 7 43 0 83 9 79 5 37;
0 93 8 24 3 69 7 38 5 77 2 87 4 60 6 41 1 87 9 83;
9 77 0 44 4 96 8 79 6 75 2 98 5 25 3 17 7 43 1 49;
3 76 2 35 5 28 0 95 7 95 4 61 8 35 1 7 6 9 9 10;
1 91 7 27 8 50 3 16 4 28 5 59 6 52 0 46 2 39 9 43;
1 45 7 71 2 39 0 87 8 11 6 54 3 41 9 43 5 9 4 20;
2 37 3 26 4 33 9 42 0 78 6 89 7 8 8 66 1 28 5 33;
1 74 0 69 5 84 3 27 9 81 7 45 8 69 2 94 6 78 4 96;
5 76 7 32 6 18 0 20 3 87 2 17 9 25 4 24 1 31 8 81;
9 97 8 90 5 28 7 86 0 58 1 72 2 23 6 76 3 99 4 45;
9 48 5 27 6 67 7 62 4 98 0 42 1 46 8 27 3 48 2 17;
9 80 3 19 5 28 1 12 4 94 6 63 7 98 8 50 0 80 2 50;
2 50 1 41 4 61 8 79 5 11 9 72 7 18 3 55 6 37 0 75;
9 22 5 57 4 75 2 14 7 65 3 96 1 71 0 47 8 79 6 60;
3 32 2 69 4 44 1 31 9 51 0 33 6 34 5 58 7 47 8 58;
8 66 7 40 2 17 0 62 9 38 5 8 6 15 3 29 1 44 4 97;
3 50 2 58 6 21 4 63 7 57 8 32 5 20 9 87 0 57 1 39;
4 20 6 67 1 85 2 90 7 70 0 84 8 30 5 56 3 61 9 15;

• LA32(30×10)

6 89 1 58 4 97 2 44 8 77 3 5 0 9 5 58 9 96 7 84;
7 31 2 81 9 73 4 15 1 87 5 39 8 57 0 77 3 85 6 21;
2 48 5 71 0 40 3 70 1 49 6 22 4 10 8 34 7 80 9 82;
4 11 6 72 7 62 0 55 2 17 5 75 3 7 1 91 9 35 8 47;
0 64 6 71 4 12 1 90 2 94 3 75 9 20 8 15 5 50 7 67;
2 29 6 93 3 68 5 93 1 57 8 77 0 52 9 7 4 58 7 70;
4 26 3 27 1 63 5 6 6 87 7 56 8 48 9 36 0 95 2 82;
1 8 7 76 3 76 4 30 6 84 9 78 8 41 0 36 2 36 5 15;
3 13 8 29 0 75 2 81 1 78 5 88 4 54 9 40 7 13 6 82;
0 52 2 6 3 6 5 82 6 64 9 88 8 54 4 54 7 32 1 26;
8 62 1 35 4 72 7 69 0 62 5 32 9 5 3 61 2 67 6 93;
2 78 3 11 7 82 4 7 1 72 8 64 9 90 0 85 5 88 6 63;
7 50 4 28 3 35 1 66 2 27 8 49 9 11 6 88 5 31 0 44;
4 62 5 39 0 76 2 14 6 56 3 97 1 7 7 69 9 66 8 47;
6 47 2 41 0 64 7 58 9 57 8 93 3 69 5 53 1 18 4 79;
7 76 9 81 0 76 6 61 4 77 8 26 2 74 5 22 1 58 3 78;
6 30 8 72 3 43 0 65 1 16 4 92 5 95 9 29 2 99 7 64;
1 35 3 74 5 16 4 85 0 7 2 81 6 86 8 61 9 35 7 34;
1 97 7 43 4 72 6 88 5 17 0 43 8 94 3 64 9 22 2 42;
7 99 2 84 8 99 5 98 1 20 6 31 3 74 0 92 9 23 4 89;

• LA31(30×10)	• LA32(30×10)
6 29 0 82 4 18 3 38 7 21 8 50 1 23 5 84 2 45 9 41;	8 32 0 6 4 55 5 19 9 81 1 81 7 40 6 9 3 37 2 40;
3 54 9 37 6 62 5 16 0 52 8 57 4 54 2 38 7 74 1 52;	6 15 2 70 8 25 1 46 9 65 4 64 7 21 0 77 5 65 3 55;
4 79 1 61 8 11 0 81 7 89 6 89 5 57 3 68 9 81 2 30;	8 31 7 84 5 37 3 24 2 85 4 89 9 29 1 44 0 40 6 83;
9 24 1 66 4 32 3 33 8 8 2 20 6 84 0 91 7 55 5 20;	4 80 0 8 9 41 5 59 7 56 3 38 2 30 8 97 6 77 1 80;
3 54 2 64 6 83 9 40 7 8 0 7 4 19 5 56 1 39 8 7;	9 59 0 91 3 50 8 80 1 17 6 40 2 71 5 56 4 88 7 7;
1 6 4 74 0 63 2 64 9 15 6 42 7 98 8 61 5 40 3 91;	7 36 3 58 4 54 5 77 2 8 6 9 0 45 9 10 1 29 8 96;
1 80 3 75 0 26 2 87 9 22 7 39 8 24 4 75 6 44 5 6;	0 28 3 92 2 73 7 27 8 86 5 87 9 96 1 98 6 99 4 70;
5 8 3 79 6 61 1 15 0 12 7 43 8 26 9 22 2 20 4 80;	9 32 1 95 3 85 6 81 2 41 8 39 7 92 0 59 5 56 4 52;
1 36 0 63 7 10 4 22 3 96 5 40 9 5 8 18 6 33 2 62;	4 93 2 12 5 22 6 27 8 45 7 69 3 60 1 7 0 88 9 49;
4 8 8 15 2 64 3 95 1 96 6 38 7 18 9 23 5 64 0 89;	2 61 5 26 9 71 8 14 0 21 6 82 3 68 7 33 1 84 4 99;

• LA33(30×10)	• LA34(30×10)
2 38 4 75 9 12 5 97 0 76 1 29 8 14 6 66 7 44 3 12;	2 51 7 59 1 35 5 73 9 65 0 27 6 13 3 81 8 32 4 74;
0 43 5 38 1 80 3 82 2 85 4 58 6 87 8 92 9 80 7 69;	4 64 7 33 5 75 2 33 8 10 0 28 3 38 6 53 9 49 1 55;
6 48 4 8 8 66 7 7 2 14 3 41 5 61 0 43 1 84 9 5;	6 83 1 23 2 72 3 7 9 72 0 6 4 39 5 52 8 90 7 21;
5 19 3 74 6 41 4 59 8 13 2 42 9 73 7 97 1 8 0 96;	3 82 1 23 2 93 4 78 6 88 7 53 9 28 8 65 5 21 0 61;
3 75 5 5 2 70 8 42 7 23 6 55 1 48 9 38 4 37 0 7;	4 41 6 12 9 12 3 77 1 70 7 24 0 81 5 73 2 62 8 6;
2 72 7 31 3 95 0 79 4 25 1 56 8 9 9 60 5 73 6 43;	4 98 3 28 6 42 9 72 0 15 8 15 5 94 2 33 1 51 7 99;
9 31 3 78 6 16 4 94 7 86 5 21 0 97 8 53 1 7 2 64;	0 32 8 22 9 96 4 15 6 78 3 31 5 7 1 94 2 23 7 86;
3 86 2 65 6 59 8 44 1 33 7 85 0 61 5 32 9 63 4 30;	7 93 2 97 3 43 5 73 0 24 8 68 9 88 1 42 4 35 6 72;
4 11 5 61 9 84 3 16 7 90 1 30 0 60 8 93 2 41 6 45;	2 14 0 44 8 13 5 67 1 63 3 49 7 5 4 17 6 85 9 66;
5 11 2 28 0 32 7 36 8 31 4 47 3 20 6 52 9 35 1 49;	7 82 9 15 3 72 4 26 0 8 1 68 6 21 8 45 2 99 5 27;
5 17 3 34 6 49 1 84 0 85 8 20 7 74 9 68 4 10 2 77;	4 93 6 23 0 51 8 54 3 49 1 96 2 56 9 36 5 53 7 52;
8 71 5 7 3 29 1 85 4 76 6 59 2 17 0 17 9 13 7 48;	8 60 0 14 4 70 9 55 1 23 5 83 3 38 2 24 7 37 6 48;
1 39 9 16 4 39 6 87 7 11 3 32 2 15 0 19 5 64 8 43;	0 62 7 15 8 69 9 23 1 82 6 26 4 45 5 33 3 12 2 37;
5 33 8 82 2 92 1 83 6 32 3 99 9 99 4 91 0 8 7 57;	6 72 1 9 7 15 5 28 8 92 9 12 0 59 3 64 4 87 2 73;
7 7 0 48 9 62 4 88 6 21 5 39 8 27 3 91 1 38 2 69;	0 50 1 14 7 90 5 46 3 71 4 48 2 80 9 61 8 24 6 44;
9 64 8 45 3 24 7 80 2 67 4 18 6 38 0 88 5 80 1 44;	0 22 9 94 5 16 3 73 2 54 8 54 4 46 1 97 6 61 7 75;
2 15 3 72 4 40 7 21 3 52 0 51 9 59 1 24 6 47 5 43;	9 55 3 67 6 77 4 30 7 6 1 32 8 47 5 93 2 6 0 40;
4 77 7 43 1 40 2 31 8 76 6 20 5 88 3 70 9 5 0 32;	1 30 3 98 7 79 0 22 6 79 2 7 8 36 9 36 5 9 4 92;
2 14 7 58 9 85 5 64 1 26 6 94 0 32 3 49 8 80 4 47;	8 37 7 72 2 52 4 31 1 82 9 54 5 7 6 82 3 73 0 49;
9 23 1 11 8 34 4 75 7 79 3 26 2 96 0 5 6 9 5 59;	1 73 3 83 7 45 2 76 4 43 9 29 0 35 5 92 8 39 6 28;
0 75 2 20 8 10 3 66 6 43 7 37 1 9 9 83 4 68 5 52;	2 58 0 26 1 48 8 52 7 34 6 96 5 70 4 98 3 80 9 94;
8 54 1 26 4 79 7 88 6 84 0 6 2 54 9 59 3 28 5 42;	1 70 8 23 5 26 4 14 6 90 2 93 3 21 0 42 7 18 9 36;
4 56 9 29 3 36 0 40 5 86 8 68 2 69 7 23 5 62 1 16;	4 28 6 76 7 25 0 17 1 84 2 67 8 87 3 43 9 88 5 84;
7 53 1 5 6 17 9 59 2 59 8 78 3 64 0 82 4 13 5 12;	7 30 3 91 8 52 4 80 0 21 5 8 9 37 2 15 6 12 1 92;
9 7 6 62 7 90 5 83 1 85 3 69 0 16 4 81 2 58 8 66;	1 28 4 7 7 46 6 92 2 77 3 15 9 69 8 54 0 47 5 39;
7 24 2 65 1 69 5 42 9 82 6 82 0 83 3 46 8 72 4 33;	9 50 5 44 2 64 8 38 4 93 6 33 7 75 0 41 1 24 3 5;
1 10 8 27 7 43 5 20 4 71 9 65 2 73 6 99 0 24 3 64;	7 94 0 17 6 87 2 21 8 92 9 28 1 61 4 63 3 34 5 77;
9 35 3 92 0 38 5 35 7 30 8 45 2 8 4 82 1 34 6 21;	3 72 8 98 9 5 4 28 2 9 5 95 6 64 1 43 0 50 7 96;
5 23 7 84 9 7 4 85 8 60 1 15 2 52 6 94 3 83 0 6;	0 85 2 85 8 39 1 98 7 24 3 71 5 60 4 55 9 22 6 35;
2 70 6 29 8 27 9 80 4 6 7 39 1 79 0 28 3 66 5 66;	3 78 6 49 2 46 1 11 0 90 5 20 9 34 7 6 4 70 8 74;

- LA35(30×10)

0 66 2 84 3 26 7 29 9 94 6 98 8 7 5 98 1 45 4 43;
3 32 0 97 6 55 2 88 8 93 9 88 1 20 4 50 7 17 5 5;
4 43 3 68 8 47 9 68 1 57 6 20 5 81 2 60 7 94 0 62;
1 57 5 40 0 78 6 9 2 49 9 17 3 32 4 30 8 87 7 77;
0 52 4 30 3 48 5 48 1 26 9 17 6 93 8 97 7 49 2 89;
7 95 0 33 1 5 6 17 5 70 3 57 4 34 2 61 8 62 9 39;
7 97 5 92 1 31 8 5 2 79 4 5 3 67 0 5 9 78 6 60;
2 79 4 6 7 20 8 45 6 34 3 24 9 26 5 68 1 16 0 46;
7 58 9 50 2 19 8 93 6 49 3 25 5 85 4 50 0 93 1 26;
9 81 6 71 5 7 1 39 2 16 8 42 0 71 4 84 3 56 7 99;
8 9 0 86 9 6 3 71 6 97 5 85 4 16 2 42 7 81 1 81;
4 72 3 24 0 30 8 56 2 43 1 61 7 82 6 40 5 59 9 43;
9 43 1 13 6 70 7 93 0 95 8 12 4 15 2 78 5 97 3 14;
0 14 6 26 1 71 3 46 8 80 5 31 4 37 9 27 7 92 2 67;
2 12 0 43 5 96 6 7 3 45 7 20 1 13 9 29 4 60 8 33;
1 78 5 50 6 84 0 42 8 84 4 30 9 76 2 57 7 87 3 59;
4 49 7 50 1 15 8 13 0 93 6 50 9 32 5 59 3 10 2 35;
1 25 0 47 7 60 8 33 4 53 5 37 9 73 2 22 3 87 6 79;
0 84 6 83 1 71 5 68 9 89 8 11 3 60 4 50 2 33 7 97;
1 14 0 38 6 88 5 5 4 77 7 92 8 24 2 73 9 52 3 71;
7 62 9 19 6 38 3 15 8 64 2 64 4 8 1 61 0 19 5 33;
2 33 5 46 4 74 0 56 6 84 9 83 8 19 7 8 3 32 1 97;
4 50 3 71 6 50 2 97 9 8 0 17 7 19 8 92 5 54 1 52;
8 32 1 79 3 97 5 38 9 49 4 76 6 76 0 56 2 78 7 54;
5 13 3 5 2 25 0 86 1 95 9 28 6 78 8 24 7 10 4 39;
7 48 2 59 0 20 9 7 5 31 6 97 1 89 4 32 3 25 8 41;
5 87 0 18 9 48 2 43 1 30 6 97 7 47 8 65 3 69 4 27;
6 71 5 20 8 20 1 78 3 39 0 17 7 50 2 44 9 42 4 38;
0 50 9 42 3 72 5 7 1 77 7 58 4 78 2 89 6 70 8 36;
3 32 9 95 2 13 0 73 6 97 8 24 4 49 5 57 1 68 7 94;

- LA36(15×15)

4 21 3 55 6 71 14 98 10 12 2 34 9 16 1 21 0 53 7 26 8 52 5 95 12 31 11 42 13 39;
11 54 4 83 1 77 7 64 8 34 14 79 12 43 0 55 3 77 6 19 9 37 5 79 10 92 13 62 2 66;
9 83 5 77 2 87 7 38 4 60 12 98 0 93 13 17 6 41 10 44 3 69 11 49 8 24 1 87 14 25;
5 77 0 96 9 28 6 7 4 95 13 35 7 35 8 76 11 9 12 95 2 43 1 75 10 61 14 10 3 79;
10 87 4 28 8 50 2 59 0 46 11 45 14 9 9 43 6 52 7 27 1 91 13 41 3 16 5 59 12 39;
0 20 2 71 4 78 13 66 3 14 12 8 14 42 6 28 1 54 9 33 11 89 8 26 7 37 10 33 5 43;
8 69 4 96 12 17 0 69 7 45 11 31 6 78 10 20 3 27 13 87 1 74 5 84 14 76 2 91 9 81;
4 58 13 90 11 76 3 81 7 23 9 28 1 18 2 32 12 86 8 99 14 97 0 24 10 45 6 72 5 25;
5 27 1 46 6 67 8 27 13 19 10 80 2 17 3 48 7 62 11 12 14 28 4 98 0 42 9 48 12 50;
11 37 5 80 4 75 8 55 7 50 0 94 9 14 6 41 14 72 3 50 10 61 13 79 2 98 12 18 1 63;

- LA36(15×15)

7 65 3 96 0 47 4 75 12 69 14 58 10 33 1 71 9 22 13 32 5 57 8 79 2 14 11 31 6 60;
1 34 2 47 3 58 5 51 4 62 6 44 9 8 7 17 10 97 8 29 11 15 13 66 12 40 0 44 14 38;
3 50 7 57 13 61 5 20 11 85 12 90 2 58 4 63 10 84 1 39 9 87 6 21 14 56 8 32 0 57;
9 84 7 45 5 15 14 41 10 18 4 82 11 29 2 70 1 67 3 30 13 50 6 23 0 20 12 21 8 38;
9 37 10 81 11 61 14 57 8 57 0 52 7 74 6 62 12 30 1 52 2 38 13 68 4 54 3 54 5 16;

- LA37(15×15)

5 19 6 64 11 73 9 13 2 84 14 88 3 85 10 41 12 53 13 80 1 66 7 46 8 59 4 25 0 62;
1 67 3 74 7 41 2 57 14 52 0 14 9 64 8 84 6 78 5 47 13 28 4 84 10 63 12 26 11 46;
6 97 8 95 0 64 9 38 10 59 12 95 2 17 11 65 13 93 3 10 5 73 1 11 4 85 14 46 7 67;
10 23 12 49 3 32 4 66 2 43 0 60 8 41 7 61 13 70 9 49 11 17 6 90 1 85 14 99 5 85;
9 98 8 57 3 73 6 9 0 73 7 7 1 98 4 13 13 41 5 40 11 85 10 37 2 68 14 79 12 17;
11 66 7 53 5 86 6 40 0 14 3 19 13 96 4 95 2 54 10 84 12 97 8 16 14 52 1 76 9 87;
4 77 2 55 9 42 5 74 14 91 13 33 10 16 12 54 0 18 3 87 7 60 8 13 6 33 1 33 11 61;
6 41 5 39 11 82 9 64 14 47 10 28 7 78 13 49 1 79 4 58 2 92 3 79 12 6 0 69 8 76;
11 21 5 42 9 91 2 28 0 52 6 88 12 76 13 86 10 23 1 35 7 52 4 91 3 47 14 82 8 24;
11 42 1 93 3 95 13 45 9 28 14 77 0 84 10 8 7 45 4 70 5 37 6 86 12 64 8 67 2 38;
4 97 12 81 1 58 7 84 5 58 0 9 11 87 3 5 2 44 13 85 6 89 10 77 9 96 14 39 8 77;
12 80 1 21 10 10 5 73 8 70 6 49 2 31 13 34 4 40 11 22 0 15 14 82 3 57 9 71 7 48;
2 17 7 62 5 75 9 35 1 91 14 50 3 7 10 64 13 75 12 94 0 55 6 72 8 47 4 11 11 90;
11 93 6 57 1 71 12 70 9 93 5 20 3 15 13 77 10 58 0 12 2 67 8 68 14 7 7 29 4 52;
13 76 3 27 4 26 9 36 11 8 10 36 0 95 8 48 2 82 6 87 5 6 1 63 7 56 12 36 14 15;

- LA38(15×15)

1 26 12 67 0 72 6 74 14 13 8 43 4 30 3 19 10 23 11 85 5 98 13 43 2 38 7 8 9 75;
14 42 0 39 4 55 12 46 1 19 8 93 9 80 5 26 10 7 6 50 11 57 3 73 2 9 7 61 13 72;
3 96 4 99 12 34 6 60 7 43 14 7 13 12 8 11 11 70 10 43 0 91 1 68 9 11 5 68 2 72;
14 63 11 45 4 49 1 74 8 27 0 30 9 72 7 9 12 99 13 60 5 69 6 69 2 84 3 40 10 59;
2 91 0 75 9 98 3 17 10 72 13 31 11 9 14 98 7 50 5 37 4 8 8 65 1 90 12 91 6 71;
11 35 6 80 4 39 3 62 14 74 5 72 10 35 9 25 1 49 8 52 7 63 2 90 13 21 12 47 0 38;
14 19 7 57 10 24 13 91 3 50 0 5 11 49 12 18 9 58 5 24 8 52 1 88 2 68 6 20 4 53;
7 77 14 72 5 35 11 90 4 68 6 18 3 9 0 33 8 60 10 18 12 10 13 60 1 38 2 99 9 15;
13 6 8 86 2 40 9 79 12 92 11 23 5 89 10 95 6 91 7 72 0 80 1 60 3 56 4 51 14 23;
1 46 6 28 5 34 11 77 4 47 0 10 14 49 8 77 10 48 7 24 12 8 2 72 13 55 9 29 3 40;
10 22 4 89 12 79 0 7 9 15 1 6 11 30 6 38 5 11 8 52 3 20 7 5 14 9 2 20 13 28;
5 73 14 56 2 37 3 22 13 25 6 58 1 8 7 93 4 88 8 17 12 9 11 69 10 71 9 85 0 55;
9 85 14 58 3 46 8 64 2 49 6 37 1 33 4 30 5 26 0 20 13 74 10 77 12 99 11 56 7 21;
10 17 3 24 4 89 5 15 11 60 1 42 8 98 2 64 13 92 0 63 7 52 12 54 6 75 14 23 9 38;
3 8 5 17 11 56 7 93 14 26 9 62 6 7 10 88 0 97 1 7 2 43 8 29 13 35 12 87 4 57;

• LA39(15×15)

10 51 14 43 7 80 4 18 6 38 3 24 2 67 12 15 11 24 13 72 8 45 5 80 9 64 1 44 0 88;
6 40 9 88 10 77 5 59 11 20 3 52 8 70 0 40 4 32 13 76 12 43 7 31 2 21 14 5 1 47;
0 32 3 49 10 5 5 64 7 58 8 80 6 94 11 11 1 26 13 26 14 59 9 85 4 47 12 96 2 14;
5 23 6 9 0 75 12 37 11 43 2 79 4 75 3 34 7 20 13 10 14 83 10 68 9 52 8 66 1 9;
12 69 9 59 3 28 14 62 13 36 1 26 6 84 11 16 8 54 5 42 2 54 0 6 10 40 7 88 4 79;
13 78 12 53 11 17 5 29 4 82 2 23 9 12 8 64 1 86 7 59 6 5 3 68 14 59 10 13 0 56;
10 83 13 46 9 7 12 65 11 69 6 62 0 16 2 58 8 66 5 83 7 90 14 42 4 81 3 69 1 85;
7 73 10 71 8 64 6 10 9 20 11 99 4 24 14 65 5 82 3 72 12 43 1 82 13 27 2 24 0 33;
4 82 1 34 3 92 2 8 0 38 8 45 6 21 5 35 12 52 9 35 11 15 14 23 10 6 13 83 7 30;
2 84 5 7 9 66 10 6 4 28 13 27 6 79 7 70 0 85 1 94 3 60 14 80 12 39 8 66 11 29;
3 44 6 58 13 14 8 65 1 72 5 14 12 52 4 21 9 25 0 5 11 51 7 61 14 55 10 42 2 36;
14 43 10 72 5 78 11 12 12 17 0 46 9 27 6 51 2 63 1 79 8 79 7 91 4 49 13 26 3 93;
7 49 0 49 4 71 5 78 9 44 10 41 12 91 13 84 8 91 6 21 11 47 14 28 3 61 2 70 1 93;
3 25 4 85 0 66 2 45 10 95 12 21 8 84 5 24 9 53 7 67 6 91 11 11 13 32 1 30 14 89;
3 92 7 93 0 99 1 40 10 37 12 69 5 66 6 57 14 22 9 44 8 73 13 97 11 18 2 69 4 41;

• LA40(15×15)

9 65 10 28 4 74 12 33 2 51 14 75 5 73 8 32 6 13 3 81 1 35 7 59 13 38 11 55 0 27;
0 64 1 53 11 83 2 33 4 6 9 52 14 72 8 7 13 90 12 21 6 23 3 10 10 39 5 49 7 72;
14 73 3 82 1 23 12 62 6 88 5 21 8 65 11 70 7 53 10 81 2 93 13 77 0 61 9 28 4 78;
1 12 6 51 7 33 4 15 14 72 10 98 9 94 5 12 11 42 2 24 13 15 8 28 3 6 12 99 0 41;
12 97 5 7 9 96 4 15 14 73 13 43 0 32 8 22 11 42 1 94 2 23 7 86 6 78 10 24 3 31;
1 72 5 88 2 93 13 13 4 44 14 66 6 63 7 14 9 67 10 17 11 85 0 35 3 68 12 5 8 49;
9 15 7 82 6 21 14 53 3 72 13 49 2 99 4 26 12 56 8 45 1 68 10 51 0 8 5 27 11 96;
3 54 7 24 4 14 8 38 5 36 2 52 14 55 12 37 11 48 0 93 13 60 10 70 1 23 6 23 9 83;
3 12 8 69 6 26 9 23 14 28 1 82 5 33 4 45 13 64 7 15 11 9 12 73 10 59 2 37 0 62;
0 87 5 12 7 80 4 50 10 48 12 90 1 72 13 24 6 14 8 71 11 44 9 46 2 15 14 61 3 92;
2 54 0 22 6 61 4 46 3 73 5 16 12 6 9 94 14 93 13 67 8 54 7 75 11 32 10 40 1 97;
10 92 14 36 4 22 9 9 3 47 1 77 12 79 13 36 6 30 8 98 11 79 7 7 5 55 2 6 0 30;
0 49 13 83 3 73 6 82 1 82 14 92 11 73 4 31 10 35 9 54 5 7 8 37 7 72 2 52 12 76;
10 98 12 34 13 52 4 26 1 28 3 39 8 80 5 29 9 70 0 43 6 48 7 58 2 45 14 94 11 96;
1 70 10 17 6 90 12 67 4 14 8 23 3 21 7 18 13 43 11 84 5 26 9 36 2 93 14 84 0 42;

附录 1.3　ABZ 类(5 个子问题)

• ABZ5(10×10)	• ABZ6(10×10)
4 88 8 68 6 94 5 99 1 67 2 89 9 77 7 99 0 86 3 92;	7 62 8 24 5 25 3 84 4 47 6 38 2 82 0 93 9 24 1 66;
5 72 3 50 6 69 4 75 2 94 8 66 0 92 1 82 7 94 9 63;	5 47 2 97 8 92 9 22 1 93 4 29 7 56 3 80 0 78 6 67;
9 83 8 61 0 83 1 65 6 64 5 85 7 78 4 85 2 55 3 77;	1 45 7 46 6 22 2 26 9 38 0 69 4 40 3 33 8 75 5 96;
7 94 2 68 1 61 4 99 3 54 6 75 5 66 0 76 9 63 8 67;	4 85 8 76 5 68 9 88 3 36 6 75 2 56 1 35 0 77 7 85;
3 69 4 88 9 82 8 95 0 99 2 67 6 95 5 68 7 67 1 86;	8 60 9 20 7 25 3 63 4 81 0 52 1 30 5 98 6 54 2 86;
1 99 4 81 5 64 6 66 8 80 2 80 7 69 9 62 3 79 0 88;	3 87 9 73 5 51 2 95 4 65 1 86 6 22 8 58 0 80 7 65;
7 50 1 86 4 97 3 96 0 95 8 97 2 66 5 99 6 52 9 71;	5 81 2 53 7 57 6 71 9 81 0 43 4 26 8 54 3 58 1 69;
4 98 6 73 3 82 2 51 1 71 5 94 7 85 0 62 8 95 9 79;	4 20 6 86 5 21 8 79 9 62 2 34 0 27 1 81 7 30 3 46;
0 94 6 71 3 81 7 85 1 66 2 90 4 76 5 58 8 93 9 97;	9 68 6 66 5 98 8 86 7 66 0 56 3 82 1 95 4 47 2 78;
3 50 0 59 1 82 8 67 7 56 9 96 6 58 4 81 5 59 2 96;	0 30 3 50 7 34 2 58 1 77 5 34 8 84 4 40 9 46 6 44;

• ABZ7(20×15)
2 24 3 12 9 17 4 27 0 21 6 25 8 27 7 26 1 30 5 31 11 18 14 16 13 39 10 19 12 26;
6 30 3 15 12 20 11 19 1 24 13 15 10 28 2 36 5 26 7 15 0 11 8 23 14 20 9 26 4 28;
6 35 0 22 13 23 7 32 2 20 3 12 12 19 10 23 9 17 1 14 5 16 11 29 8 16 4 22 14 22;
9 20 6 29 1 19 7 14 12 33 4 30 0 32 5 21 11 29 10 24 14 25 2 29 3 13 8 20 13 18;
11 23 13 20 1 28 6 32 7 16 5 18 8 24 9 23 3 24 10 34 2 24 0 24 14 28 12 15 4 18;
8 24 11 19 14 21 1 33 7 34 6 35 5 40 10 36 3 23 2 26 4 15 9 28 13 38 12 13 0 25;
13 27 3 30 6 21 8 19 12 12 4 27 2 39 9 13 14 12 5 36 10 21 11 17 1 29 0 17 7 33;
5 27 4 19 6 29 9 20 3 21 10 40 8 14 14 39 13 39 2 27 1 36 12 12 11 37 7 22 0 13;
13 32 11 29 8 24 3 27 5 40 4 21 9 26 0 27 14 27 6 16 2 21 10 13 7 28 12 28 1 32;
12 35 1 11 5 39 14 18 7 23 0 34 3 24 13 11 8 30 11 31 4 15 10 15 2 28 9 26 6 33;
10 28 5 37 12 29 1 31 7 25 8 13 14 14 4 20 3 27 9 25 13 31 11 14 6 25 2 39 0 36;
0 22 11 25 5 28 13 35 4 31 8 21 9 20 14 19 2 29 7 32 10 18 1 18 3 11 12 17 6 15;
12 39 5 32 2 36 8 14 3 28 13 37 0 38 6 20 7 19 11 12 14 22 1 36 4 15 9 32 10 16;
8 28 1 29 14 40 12 23 4 34 5 33 6 27 10 17 0 20 7 28 11 21 2 21 13 20 9 33 3 27;
9 21 14 34 3 30 12 38 0 11 11 16 2 14 5 14 1 34 8 33 4 23 13 40 10 12 6 23 7 27;
9 13 14 40 7 36 4 17 0 13 5 33 8 25 13 24 10 23 3 36 2 29 1 18 11 13 6 33 12 13;
3 25 5 15 2 28 12 40 7 39 1 31 8 35 6 31 11 36 4 12 10 33 14 19 9 16 13 27 0 21;
12 22 10 14 0 12 2 20 5 12 1 18 11 17 8 39 14 31 3 31 7 32 9 20 13 29 4 13 6 26;
5 18 10 30 7 38 14 22 13 15 11 20 9 16 3 17 1 12 2 13 12 40 6 17 8 30 4 38 0 13;
9 31 8 39 12 27 1 14 5 33 3 31 11 22 13 36 0 16 7 11 14 14 4 29 6 28 2 22 10 17;

- ABZ8(20×15)

0 19 9 33 2 32 13 18 10 39 8 34 6 25 4 36 11 40 12 33 1 31 14 30 3 34 5 26 7 13;
9 11 10 22 14 19 5 12 4 25 6 38 0 29 7 39 13 19 11 22 1 23 3 20 2 40 12 19 8 26;
3 25 8 17 11 24 13 40 10 32 14 16 5 39 9 19 0 24 1 39 4 17 2 35 7 38 6 20 12 31;
14 22 3 36 2 34 12 17 4 30 13 12 1 13 6 25 9 12 7 18 10 31 0 39 5 40 8 26 11 37;
12 32 14 15 1 35 7 13 8 32 11 23 6 22 4 21 0 38 2 38 3 40 10 31 5 11 13 37 9 16;
10 23 12 38 8 11 14 27 9 11 6 25 5 14 4 12 2 27 11 26 7 29 3 28 13 21 0 20 1 30;
6 39 8 38 0 15 12 27 10 22 9 27 2 32 4 40 3 12 13 20 14 21 11 22 5 17 7 38 1 27;
11 11 13 24 10 38 8 15 9 19 14 13 5 30 0 26 2 29 6 33 12 21 1 15 3 21 4 28 7 33;
8 20 6 17 5 26 3 34 9 23 0 16 2 18 4 35 12 24 10 16 11 26 7 12 14 13 13 27 1 19;
1 18 7 37 14 27 9 40 5 40 6 17 8 22 3 17 10 30 0 38 4 21 12 32 11 24 13 24 2 30;
11 19 0 22 13 36 6 18 5 22 3 17 14 35 10 34 7 23 8 19 2 29 1 22 12 17 4 33 9 39;
6 32 3 22 12 24 5 13 4 13 1 11 0 11 13 25 8 13 2 15 10 33 11 17 14 16 9 38 7 24;
14 16 13 16 1 37 8 25 2 26 3 11 9 34 4 14 0 20 6 36 12 12 5 29 10 25 7 32 11 12;
8 20 10 24 11 27 9 38 5 34 12 39 7 33 4 37 2 31 13 15 14 34 3 33 6 26 1 36 0 14;
8 31 0 17 9 13 1 21 10 17 7 19 13 14 3 40 5 32 11 25 2 34 14 23 6 13 12 40 4 26;
8 38 12 17 3 14 13 17 4 12 1 35 6 35 0 19 10 36 7 19 9 29 2 31 5 26 11 35 14 37;
14 20 3 16 0 33 10 14 5 27 7 31 8 16 6 31 12 28 9 37 4 37 2 29 11 38 1 30 13 36;
11 18 3 37 14 16 6 15 8 14 12 11 13 32 5 12 1 11 10 29 7 19 4 12 9 18 2 26 0 39;
11 11 2 11 12 22 9 35 14 20 7 31 4 19 3 39 5 28 6 33 10 34 1 38 0 20 13 17 8 28;
2 12 12 25 5 23 8 21 6 27 9 30 14 23 11 39 3 26 13 34 7 17 1 24 4 12 0 19 10 36;

- ABZ9(20×15)

6 14 5 21 8 13 4 11 1 11 14 35 13 20 11 17 10 18 12 11 2 23 3 13 0 15 7 11 9 35;
1 35 5 31 0 13 3 26 6 14 9 17 7 38 12 20 10 19 13 12 8 16 4 34 11 15 14 12 2 14;
0 30 4 35 2 40 10 35 6 30 14 23 8 29 13 37 7 38 3 40 9 26 12 11 1 40 11 36 5 17;
7 40 5 18 4 12 8 23 0 23 9 14 13 16 12 14 10 23 3 12 6 16 14 32 1 40 11 25 2 29;
2 35 3 15 12 31 11 28 6 32 4 30 10 27 7 29 0 38 13 11 1 23 14 17 5 27 9 37 8 29;
5 33 3 33 6 19 12 40 10 19 0 33 13 26 2 31 11 28 7 36 4 38 1 21 14 25 9 40 8 35;
13 25 0 32 11 33 12 18 4 32 6 28 5 15 3 35 9 14 2 34 7 23 10 32 1 17 14 26 8 19;
2 16 12 33 9 34 11 30 13 40 8 12 14 26 5 26 6 15 3 21 1 40 4 32 0 14 7 30 10 35;
2 17 10 16 14 20 6 24 8 26 3 36 12 22 0 14 13 11 9 20 7 23 1 29 11 23 4 15 5 40;
4 27 9 37 3 40 11 14 13 25 7 30 0 34 2 11 5 15 12 32 1 36 10 12 14 28 8 31 6 23;
13 25 0 22 3 27 8 14 5 25 6 20 14 18 7 14 1 19 2 17 4 27 9 22 12 22 11 27 10 21;
14 34 10 15 0 22 3 29 13 34 6 40 7 17 2 32 12 20 5 39 4 31 11 16 1 37 8 33 9 13;
6 12 12 27 4 17 2 24 8 11 5 19 14 11 3 17 9 25 1 11 11 31 13 33 7 31 10 12 0 22;
5 22 14 15 0 16 8 32 7 20 4 22 9 11 13 19 1 30 12 33 6 29 11 18 3 31 10 32 2 18;
5 27 3 26 10 28 6 37 4 18 12 12 11 11 13 26 7 27 9 40 14 19 1 24 2 18 0 12 8 34;
8 15 5 28 9 25 6 32 1 13 7 38 11 11 2 34 4 25 0 20 10 32 3 23 12 14 14 16 13 20;
1 15 4 13 8 37 3 14 10 22 5 24 12 20 7 22 9 34 14 22 11 19 13 32 0 29 2 13 6 35;
7 36 5 33 13 28 9 20 10 30 4 33 14 29 0 34 3 22 11 12 6 30 8 12 1 35 2 13 12 35;
14 26 11 31 5 35 2 38 13 19 10 35 4 27 8 29 3 39 9 13 6 14 7 26 0 17 1 22 12 15;
1 36 7 34 11 33 8 17 14 38 6 39 5 16 3 27 13 29 2 16 0 16 4 19 9 40 12 35 10 39;

附录 1.4 ORB 类(10 个子问题)

• ORB01(10×10)	• ORB02(10×10)
0 72 1 64 2 55 3 31 4 53 5 95 6 11 7 52 8 6 9 84; 0 61 3 27 4 88 2 78 1 49 5 83 8 91 6 74 7 29 9 87; 0 86 3 32 1 35 2 37 5 18 4 48 6 91 7 52 9 60 8 30; 0 8 1 82 4 27 3 99 6 74 5 9 2 33 9 20 7 59 8 98; 1 50 0 94 5 43 3 62 4 55 7 48 2 5 8 36 9 47 6 36; 0 53 6 30 2 7 3 12 1 68 8 87 4 28 9 70 7 45 5 7; 2 29 3 96 0 99 1 14 4 34 7 14 5 7 6 76 8 57 9 76; 2 90 0 19 3 87 4 51 1 84 5 45 9 84 6 58 7 81 8 96; 2 97 1 99 4 93 0 38 7 13 5 96 3 40 9 64 6 32 8 45; 2 44 0 60 8 29 3 5 6 74 1 85 4 34 7 95 9 51 5 47;	0 72 1 54 2 33 3 86 4 75 5 16 6 96 7 7 8 99 9 76; 0 16 3 88 4 48 8 52 9 60 6 29 7 18 5 89 2 80 1 76; 0 47 7 11 3 14 2 56 6 16 4 83 1 10 5 61 8 24 9 58; 0 49 1 31 3 17 8 50 5 63 2 35 4 65 7 23 6 50 9 29; 0 55 6 6 1 28 3 96 5 86 2 99 9 14 7 70 8 64 4 24; 4 46 0 23 5 70 8 19 2 54 3 22 9 85 7 87 5 79 1 93; 4 76 3 60 0 76 9 98 2 76 1 50 8 86 7 14 6 27 5 57; 4 93 6 27 9 57 3 87 8 86 2 54 7 24 5 49 0 20 1 47; 2 28 6 11 8 78 7 85 4 63 9 81 3 10 1 9 5 46 0 32; 2 22 9 76 5 89 8 13 6 88 3 10 7 75 4 98 1 78 0 17;

• ORB03(10×10)	• ORB04(10×10)
0 96 1 69 2 25 3 5 4 55 5 15 6 88 7 11 8 17 9 82; 0 11 1 48 2 67 3 38 4 18 7 24 6 62 5 92 9 96 8 81; 2 67 1 63 0 93 4 85 3 25 5 72 6 51 7 81 8 58 9 15; 2 30 1 35 0 27 4 82 3 44 7 92 6 25 5 49 9 28 8 77; 1 53 0 83 4 73 3 26 2 77 6 33 5 92 9 99 8 38 7 38; 1 20 0 44 4 81 3 88 2 66 6 70 5 91 9 37 8 55 7 96; 1 21 2 93 4 22 0 56 3 34 6 40 7 53 9 46 5 29 8 63; 1 32 2 63 4 36 0 26 3 17 5 85 7 15 8 55 9 16 6 82; 0 73 2 46 3 89 4 24 1 99 6 92 7 7 9 51 5 19 8 14; 0 52 2 20 3 70 4 98 1 23 5 15 7 81 8 71 9 24 6 81;	0 8 1 10 2 35 3 44 4 15 5 92 6 70 7 89 8 50 9 12; 0 63 8 39 3 80 5 22 2 88 1 39 9 85 6 27 7 74 4 69; 0 52 6 22 1 33 3 68 8 27 2 68 5 25 4 34 7 24 9 84; 0 31 1 85 4 55 8 80 5 58 7 11 6 69 9 56 3 73 2 25; 0 97 5 98 9 87 8 47 7 77 4 90 3 98 2 80 1 39 6 40; 1 97 5 68 0 44 9 67 2 44 8 85 3 78 6 90 7 33 4 81; 0 34 3 76 8 48 7 61 9 11 2 36 4 33 6 98 1 7 5 44; 0 44 9 5 4 85 1 51 5 58 7 79 2 95 6 48 3 86 8 73; 0 24 1 63 9 48 7 77 8 73 6 74 4 63 5 17 2 93 3 84; 0 51 2 5 4 40 9 60 1 46 5 58 8 54 3 72 6 29 7 94;

• ORB05(10×10)	• ORB06(10×10)
9 11 8 93 0 48 7 76 6 13 5 71 3 59 2 90 4 10 1 65; 8 52 9 76 0 84 7 73 5 56 4 10 6 26 2 43 3 39 1 49; 9 28 8 44 7 26 6 66 4 68 5 74 3 27 2 14 1 6 0 21; 0 18 1 58 3 62 2 46 6 25 4 6 5 60 7 28 8 80 9 30; 0 78 1 47 7 29 5 16 4 29 6 57 3 78 2 87 8 39 9 73; 9 66 8 51 3 12 7 64 5 67 4 15 6 66 2 26 1 20 0 98; 8 23 9 76 6 45 7 75 5 24 3 18 4 83 2 15 1 88 0 17; 9 56 8 83 7 80 6 16 4 31 5 93 3 30 2 29 1 66 0 28; 9 79 8 69 2 82 4 16 5 62 3 41 6 91 7 35 0 34 1 75; 0 5 1 19 2 20 3 12 4 94 5 60 6 99 7 31 8 96 9 63;	0 99 1 74 2 49 3 67 4 17 5 7 6 9 7 39 8 35 9 49; 0 49 3 67 4 82 2 92 1 62 5 84 8 45 6 30 7 42 9 71; 0 26 3 33 1 82 2 98 5 83 4 16 6 64 7 65 9 36 8 77; 0 41 1 62 4 73 3 94 6 51 5 46 2 55 9 31 7 64 8 46; 1 68 0 26 5 50 3 46 4 25 7 88 2 6 8 13 9 98 6 84; 0 24 6 80 2 91 3 55 1 48 8 99 4 72 9 91 7 84 5 12; 2 16 3 13 0 9 1 58 4 23 7 85 5 36 6 89 8 71 9 41; 2 54 0 41 3 38 4 53 1 11 5 74 9 88 6 46 7 41 8 65; 2 53 1 50 4 40 0 90 7 7 5 80 3 57 9 60 6 91 8 47; 2 45 0 59 8 81 3 99 6 71 1 19 4 75 7 77 9 94 5 95;

• ORB07(10×10)	• ORB08(10×10)
0 32 1 14 2 15 3 37 4 18 5 43 6 19 7 27 8 28 9 31;	0 55 1 74 2 45 3 23 4 76 5 19 6 18 7 61 8 44 9 11;
0 8 3 12 4 49 8 24 9 52 6 19 7 23 5 19 2 17 1 32;	0 63 1 43 2 51 3 18 4 42 7 11 6 29 5 52 9 29 8 88;
0 25 7 19 3 27 2 45 6 21 4 15 1 13 5 16 8 43 9 19;	2 88 1 31 0 47 4 10 3 62 5 60 6 58 7 29 8 52 9 92;
0 24 1 18 3 41 8 29 5 14 2 17 4 23 7 15 6 18 9 23;	2 16 1 71 0 55 4 55 3 9 7 49 6 83 5 54 9 7 8 57;
0 27 6 29 1 39 3 21 5 15 2 15 9 25 7 26 8 44 4 20;	1 7 0 41 4 92 3 94 2 46 6 79 5 34 9 38 8 8 7 18;
4 17 0 15 6 51 8 17 2 46 3 16 9 33 7 25 5 30 1 25;	1 25 0 5 4 89 3 94 2 14 6 94 5 20 9 23 8 44 7 39;
4 15 3 31 0 25 9 12 2 13 1 51 8 19 7 21 6 12 5 26;	1 24 2 21 4 47 0 40 3 94 6 71 7 89 9 75 5 97 8 15;
4 8 6 29 9 25 3 15 8 17 2 22 7 32 5 20 0 11 1 28;	1 5 2 7 4 74 0 28 3 72 5 61 7 9 8 53 9 32 6 97;
2 41 6 10 8 32 7 5 4 21 9 59 3 26 1 10 5 16 0 29;	0 34 2 52 3 37 4 6 1 94 6 6 7 56 9 41 5 5 8 16;
2 20 9 7 5 44 8 22 6 33 3 25 7 29 4 12 1 14 0 0;	0 77 2 74 3 82 4 10 1 29 5 15 7 51 8 65 9 37 6 21;

• ORB09(10×10)	• ORB10(10×10)
0 36 1 96 2 86 3 7 4 20 5 9 6 39 7 79 8 82 9 24;	9 66 8 13 0 93 7 91 6 14 5 70 3 99 2 53 4 86 1 16;
0 16 8 95 3 67 5 63 2 87 1 24 9 62 6 49 7 92 4 16;	8 34 9 99 0 62 7 65 5 62 4 64 6 21 2 12 3 9 1 75;
0 65 6 71 1 9 3 67 8 70 2 48 5 49 4 66 7 5 9 96;	9 12 8 26 7 64 6 92 4 67 5 28 3 66 2 83 1 38 0 58;
0 50 1 31 4 6 8 13 5 98 7 97 6 93 9 30 3 34 2 83;	0 77 1 73 3 82 2 75 6 84 4 19 5 18 7 89 8 8 9 73;
0 99 5 7 9 55 8 78 7 68 4 81 3 90 2 75 1 66 6 40;	0 34 1 74 7 48 5 44 4 92 6 40 3 60 2 62 8 22 9 67;
1 42 5 11 0 5 9 39 2 10 8 30 3 39 6 50 7 20 4 51;	9 8 8 85 3 58 7 97 5 92 4 89 6 75 2 77 1 95 0 5;
0 38 3 68 8 86 7 77 9 32 2 89 4 37 6 53 1 43 5 89;	8 52 9 43 6 5 7 78 5 12 3 62 4 21 2 80 1 60 0 31;
0 19 9 11 4 37 1 41 5 72 7 7 2 52 6 31 3 68 8 10;	9 81 8 23 7 23 6 75 4 78 5 56 3 51 2 39 1 53 0 96;
0 83 1 21 9 23 7 87 8 58 6 89 4 74 5 29 2 74 3 23;	9 79 8 55 2 88 4 21 5 83 3 93 6 47 7 10 0 63 1 14;
0 44 2 57 4 69 9 50 1 65 5 69 8 60 3 58 6 89 7 13;	0 43 1 63 2 83 3 29 4 52 5 98 6 54 7 39 8 33 9 23;

附录 1.5　SWV 类(20 个子问题)

• SWV01-hard(20×10)	• SWV02-harc(20×10)
3 19 2 27 1 39 4 13 0 25 8 37 9 40 5 54 7 74 6 93;	2 16 1 58 0 22 4 24 3 53 8 9 9 57 7 63 5 92 6 43;
2 69 0 30 4 1 3 4 1 64 7 71 5 2 9 81 6 31 8 8;	3 6 1 48 4 14 0 66 2 24 7 2 9 85 6 73 8 19 5 99;
4 79 3 80 0 86 2 55 1 54 8 81 6 72 7 86 5 59 9 75;	4 100 2 90 0 63 1 14 3 31 5 27 9 15 8 1 6 51 7 33;
2 76 3 15 1 26 0 17 4 30 8 44 7 91 6 83 5 52 9 68;	2 98 3 84 4 52 0 12 1 96 9 50 6 74 8 93 5 45 7 49;
4 73 3 87 1 74 0 39 2 98 9 100 5 43 8 17 7 7 6 77;	4 39 0 54 2 28 3 8 1 30 8 57 6 75 5 9 7 41 9 19;
1 63 0 49 2 16 3 55 4 9 9 73 5 61 8 34 6 82 7 46;	3 94 0 8 2 89 1 13 4 37 8 36 6 63 9 24 5 71 7 97;
0 87 1 71 4 43 3 80 2 39 7 70 8 18 6 41 9 79 5 44;	3 90 2 69 1 25 4 15 0 65 7 52 6 56 9 91 8 83 5 86;
4 70 2 22 0 73 3 62 1 64 5 25 8 19 6 69 9 41 7 28;	3 59 1 99 4 41 0 68 2 14 7 4 9 55 6 48 8 13 5 15;
3 16 0 84 1 58 4 7 2 9 5 8 6 10 7 17 8 42 9 65;	4 36 2 17 1 51 0 16 3 54 8 45 5 50 7 98 6 68 9 82;
3 8 0 10 1 3 4 41 2 3 7 40 8 56 5 53 9 96 6 13;	1 75 0 11 4 55 2 93 3 51 6 61 9 40 7 19 8 24 5 55;

• SWV01-hard(20×10)	• SWV02-hard(20×10)
4 62 1 60 3 64 2 12 0 39 5 2 7 64 6 87 9 21 8 60;	4 56 0 73 3 59 2 38 1 51 6 99 8 29 9 53 5 7 7 72;
2 66 1 71 3 23 4 75 0 78 7 74 6 35 9 24 8 23 5 50;	3 68 4 50 1 88 2 88 0 33 5 47 8 52 6 26 9 74 7 68;
1 5 3 92 4 6 0 69 2 80 7 13 5 17 9 89 6 80 8 47;	2 3 3 42 0 45 1 57 4 28 5 14 8 22 9 31 6 44 7 38;
0 82 3 84 1 24 2 47 4 93 7 85 5 34 6 73 8 28 9 91;	3 89 0 73 4 12 1 9 2 49 5 11 8 15 7 41 9 37 6 10;
4 55 0 57 3 63 2 24 1 40 7 30 6 37 5 99 8 88 9 41;	3 76 2 97 4 100 1 92 0 25 5 8 9 92 7 51 6 58 8 65;
1 75 2 47 3 68 0 7 4 78 7 80 6 2 9 23 8 49 5 50;	4 50 0 54 3 85 1 47 2 45 6 99 9 39 5 32 8 87 7 56;
0 91 4 25 2 10 1 21 3 94 8 6 7 59 5 84 9 75 6 70;	0 70 2 58 3 33 1 85 4 25 8 5 7 65 9 20 6 52 5 44;
2 85 1 31 0 94 4 94 3 11 5 21 9 7 6 61 8 50 7 93;	1 22 3 45 4 60 0 66 2 5 7 61 6 73 9 60 5 14 8 44;
1 27 0 77 4 13 2 30 3 2 5 88 7 4 9 39 6 53 8 34;	4 64 0 97 2 31 1 4 3 43 9 47 7 93 6 100 5 10 8 51;
1 34 2 12 3 31 0 24 4 24 7 16 5 6 9 88 8 81 6 11;	3 9 4 87 2 34 0 62 1 56 5 66 8 95 7 56 9 42 6 86;

• SWV03-hard(20×10)	• SWV04-hard(20×10)
2 19 0 30 1 68 4 55 3 24 8 34 7 72 5 32 9 62 6 45;	2 16 0 59 4 10 3 95 1 64 8 92 9 56 7 3 5 73 6 17;
2 63 1 11 4 65 3 16 0 67 9 95 8 23 7 82 6 52 5 53;	1 5 4 64 3 30 2 14 0 96 9 11 8 73 7 35 6 93 5 12;
2 19 4 17 1 79 3 49 0 12 7 41 9 67 8 40 6 25 5 42;	3 35 4 75 0 54 1 30 2 83 9 20 8 29 7 38 6 90 5 39;
0 42 2 71 3 27 4 95 1 19 5 48 8 100 6 31 7 25 9 38;	4 29 3 21 0 52 2 93 1 20 5 5 7 11 8 53 9 56 6 98;
3 1 1 100 4 68 0 94 2 89 5 86 7 35 9 29 8 56 6 55;	0 17 3 16 4 41 1 78 2 100 5 55 8 27 6 2 7 87 9 55;
4 93 1 53 2 4 3 48 0 57 8 99 7 67 5 86 6 80 9 60;	3 97 1 32 4 84 2 71 0 38 9 64 7 16 5 5 6 41 8 41;
4 82 1 95 2 12 0 60 3 80 8 88 7 5 6 81 9 52 5 69;	3 41 1 57 4 37 0 64 2 92 6 19 9 47 7 94 8 79 5 21;
3 79 1 31 4 63 0 28 2 64 8 63 5 29 7 75 9 18 6 33;	0 23 3 67 1 39 4 98 2 63 8 83 5 45 6 89 9 81 7 44;
4 9 1 64 2 31 0 13 3 33 9 82 6 79 5 30 7 84 8 20;	1 88 0 59 3 39 2 63 4 91 8 36 5 44 6 45 9 43 7 12;
2 14 0 56 1 95 4 34 3 13 6 16 5 44 7 45 8 62 9 86;	2 29 1 17 0 6 3 74 4 51 9 14 6 2 5 56 7 49 8 14;
4 66 3 9 2 66 1 46 0 12 5 10 7 58 6 6 8 62 9 17;	3 75 2 10 4 1 0 35 1 99 7 56 5 95 9 78 6 53 8 82;
4 89 1 52 2 37 3 74 0 7 8 43 5 95 7 89 6 21 9 66;	0 75 2 96 1 21 3 90 4 55 6 23 7 40 9 76 8 55 5 45;
1 73 3 68 2 5 4 49 0 67 9 23 7 7 5 44 8 30 6 29;	3 90 4 64 0 72 2 33 1 59 7 51 6 74 5 85 9 76 8 38;
2 21 0 68 1 88 4 75 3 64 6 6 8 72 7 66 9 66 5 56;	3 57 1 84 2 87 4 2 0 68 8 4 5 77 6 37 7 37 9 94;
1 24 4 25 2 69 0 27 3 51 9 60 8 26 6 45 5 77 7 93;	1 16 3 46 4 34 2 23 0 77 7 68 8 14 9 54 5 37 6 99;
2 19 3 17 1 82 4 75 0 34 5 67 9 89 6 91 7 13 8 35;	4 24 1 73 2 92 0 43 3 42 5 81 7 99 8 88 9 80 6 5;
4 2 0 21 3 83 1 19 2 65 6 65 8 8 9 68 7 60 5 7;	1 56 2 51 0 3 4 87 3 25 5 62 7 11 8 88 6 68 9 29;
1 63 3 49 2 4 4 2 0 50 9 99 5 27 6 68 8 46 7 89;	2 85 3 3 4 21 0 49 1 79 8 38 5 37 9 72 7 18 6 18;
0 48 4 45 3 100 2 66 1 30 6 58 7 73 9 94 5 36 8 5;	0 2 3 55 1 31 2 29 4 98 5 92 6 43 8 99 7 67 9 41;
2 36 0 53 4 56 3 57 1 77 9 7 6 59 8 8 5 15 7 23;	4 69 3 64 0 61 1 13 2 31 5 6 8 84 9 94 7 32 6 54;

- SWV05-hard(20×10)

2 19 1 30 3 80 0 84 4 14 8 51 5 73 6 91 7 81 9 71;
2 74 4 79 1 39 0 7 3 66 9 6 5 93 8 76 6 21 7 76;
4 90 3 33 1 38 2 73 0 61 8 61 7 76 5 86 9 28 6 35;
4 1 3 22 2 1 0 77 1 33 6 98 5 4 9 27 8 8 7 68;
2 63 4 5 1 95 0 7 3 50 8 46 9 28 6 70 5 60 7 34;
0 98 1 73 4 15 3 21 2 32 7 24 9 9 8 24 5 7 6 34;
3 51 4 47 2 30 1 16 0 51 5 41 6 79 7 79 9 3 8 72;
4 3 1 59 0 53 3 20 2 19 6 20 9 16 7 90 5 96 8 18;
1 34 2 55 3 97 0 93 4 90 7 81 5 63 8 11 6 1 9 51;
4 77 3 87 1 92 2 83 0 45 7 75 9 60 6 75 5 93 8 33;
0 31 2 66 1 58 4 17 3 94 5 63 7 80 9 61 6 78 8 52;
4 70 1 25 2 75 0 89 3 41 7 100 5 73 6 28 8 94 9 88;
1 67 4 62 3 12 2 55 0 62 5 58 8 66 7 73 6 55 9 1;
4 81 0 37 1 2 3 39 2 17 7 74 6 71 8 61 5 42 9 5;
3 62 0 31 4 63 2 31 1 5 9 7 7 77 8 34 6 34 5 3;
0 5 2 55 3 62 1 82 4 80 6 6 8 7 7 29 5 80 9 89;
3 26 1 50 2 58 0 22 4 68 7 12 6 9 9 31 5 90 8 87;
0 50 2 28 1 64 4 34 3 63 7 9 9 48 6 63 8 61 5 2;
0 47 2 23 1 23 4 82 3 98 7 66 6 78 8 100 9 79 5 32;
1 13 4 14 0 90 2 77 3 80 9 30 7 31 5 36 6 51 8 69;

- SWV06-hard(20×15)

1 16 6 58 2 22 4 24 5 53 3 9 0 57 10 63 8 92 12 43 7 41 13 26 14 20 9 44 11 93;
2 89 1 94 0 86 3 13 6 54 4 41 5 55 7 98 13 38 14 80 9 1 11 100 12 90 10 63 8 14;
1 26 6 96 3 32 4 75 5 9 0 57 2 39 12 54 14 28 10 8 11 30 13 57 9 75 7 9 8 41;
3 37 2 36 5 63 0 24 6 71 1 97 4 74 14 19 12 45 8 24 11 71 13 53 10 61 9 6 7 32;
3 57 0 55 1 21 5 84 2 23 6 79 4 90 11 8 14 59 10 99 9 41 12 68 8 14 13 4 7 55;
4 10 2 81 1 13 3 78 0 78 5 10 6 18 9 37 11 21 7 88 12 75 14 11 13 55 10 93 8 51;
6 100 2 52 3 54 1 37 5 26 4 74 0 87 8 13 12 88 10 94 14 73 7 55 11 68 9 50 13 88;
4 47 5 70 6 7 2 72 0 62 3 30 1 95 10 18 9 65 7 69 13 89 8 89 14 64 12 81 11 25;
6 1 1 10 0 72 3 59 4 92 5 53 2 89 14 52 7 48 8 8 13 69 10 49 9 26 12 76 11 97;
6 85 2 47 4 45 1 99 0 39 5 32 3 87 10 56 8 98 11 13 7 96 12 71 14 95 9 11 13 78;
0 17 2 21 3 87 6 41 5 41 4 31 1 96 8 17 11 95 13 29 14 3 10 71 7 64 9 97 12 31;
6 9 0 87 4 34 1 62 3 56 5 66 2 95 9 56 14 42 8 86 7 68 12 82 10 82 13 52 11 97;
3 86 1 37 2 49 0 2 6 30 5 63 4 4 14 47 8 84 10 5 13 13 9 39 12 18 7 76 11 63;
0 29 6 34 1 53 3 7 5 19 4 26 2 63 12 22 10 98 13 77 14 11 7 87 9 5 11 44 8 42;
6 44 4 91 1 91 2 58 0 77 3 51 5 14 13 1 9 17 7 55 12 40 8 95 14 31 11 54 10 37;
5 59 4 47 1 56 6 39 2 7 0 43 3 39 13 75 10 43 12 32 9 6 11 93 7 69 8 47 14 93;
4 24 1 30 3 97 6 17 0 7 2 55 5 8 7 70 10 87 8 29 12 20 13 29 11 51 9 14 14 32;
2 29 4 99 3 17 0 96 1 50 5 67 6 91 10 91 13 14 12 14 7 19 8 36 11 11 14 83 9 6;
0 7 6 60 3 31 5 76 1 23 2 83 4 30 8 73 14 76 11 17 10 53 13 9 12 72 7 89 9 24;
3 63 0 89 2 2 1 46 6 86 5 74 4 1 7 34 9 30 12 19 13 48 11 75 8 72 14 47 10 58;

• SWV07-hard(20×15)
3 92 1 49 2 93 6 48 0 1 4 52 5 57 8 16 12 6 13 6 11 19 9 96 7 27 14 76 10 60;
5 4 3 96 6 52 1 87 2 94 4 83 0 9 11 85 10 47 8 63 9 31 13 26 12 46 7 49 14 48;
1 34 6 34 4 37 2 82 0 25 5 43 3 11 9 71 14 55 7 34 11 77 12 20 8 89 10 23 13 32;
3 49 5 12 6 52 2 76 0 64 1 51 4 84 10 42 12 5 7 45 8 20 11 93 14 48 13 75 9 100;
2 35 1 1 3 15 6 49 5 78 4 80 0 99 9 88 7 24 11 20 10 100 8 28 14 71 13 1 12 7;
3 69 6 24 5 21 4 3 1 28 2 8 0 42 10 33 11 40 9 50 8 8 13 5 12 13 7 42 14 73;
0 83 4 15 2 62 6 27 5 5 1 65 3 100 14 65 10 82 7 89 13 81 9 92 8 38 11 47 12 96;
6 98 4 24 2 75 0 57 1 93 3 74 5 10 7 44 13 59 11 51 12 82 14 65 10 8 8 12 9 24;
4 55 0 44 3 47 5 75 2 81 6 30 1 42 10 100 8 81 7 29 13 31 9 47 11 34 12 77 14 92;
2 18 5 42 0 37 4 1 3 67 6 20 1 91 8 21 14 57 12 100 10 100 11 59 13 77 9 21 7 98;
3 42 1 16 4 19 6 70 2 7 0 74 5 7 12 50 9 74 8 46 14 88 13 71 10 42 7 34 11 60;
6 12 4 45 2 7 0 15 1 22 3 31 5 70 13 88 9 46 8 44 14 45 12 87 11 5 7 99 10 70;
4 51 5 39 0 50 2 9 3 23 6 28 1 49 13 5 12 17 14 40 10 30 11 62 8 65 7 84 9 12;
6 92 0 67 5 85 1 88 3 18 4 13 2 70 7 69 14 10 13 52 8 42 11 82 10 19 12 21 9 5;
4 34 0 60 1 52 5 70 2 51 6 2 3 43 10 75 11 45 8 53 12 96 13 1 14 44 7 66 9 19;
6 31 1 44 0 84 3 16 4 10 2 4 5 48 13 67 14 11 12 21 8 78 7 42 11 44 9 37 10 35;
1 20 4 40 3 37 2 68 6 42 0 11 5 6 10 44 11 43 12 17 14 3 7 77 13 100 9 82 8 5;
5 14 0 5 3 40 1 70 4 63 2 59 6 42 9 74 13 32 7 50 10 21 14 29 12 83 11 64 8 45;
6 70 0 28 3 79 4 25 5 98 2 24 1 54 12 65 13 93 10 74 7 22 9 73 11 75 8 69 14 9;
5 100 2 46 4 69 3 41 1 3 6 18 0 41 8 94 11 97 12 30 14 96 7 7 9 86 13 83 10 90;

• SWV08-hard(20×15)
3 8 4 73 2 49 5 24 6 81 1 68 0 23 12 69 8 74 10 45 11 4 14 59 9 25 7 70 13 68;
3 34 2 33 5 7 1 69 4 54 6 18 0 38 8 28 12 12 14 50 10 66 7 81 9 81 13 91 11 66;
0 8 6 20 3 52 4 83 5 18 2 82 1 68 7 50 14 54 11 6 10 73 13 48 9 20 8 93 12 99;
2 41 0 72 1 91 4 52 5 30 3 1 6 92 13 52 8 41 9 45 14 43 12 97 10 64 11 71 7 76;
0 48 1 44 5 49 6 92 3 29 2 29 4 88 14 14 10 99 8 22 13 79 9 93 12 69 11 63 7 68;
0 56 6 42 2 42 3 93 1 80 4 54 5 94 12 80 14 69 11 39 8 85 10 95 13 12 9 28 7 64;
0 90 4 75 6 9 1 46 2 91 3 93 5 93 14 77 9 63 11 50 12 82 13 74 8 67 7 72 10 76;
0 55 2 90 6 11 3 60 4 75 1 23 5 74 11 54 7 97 12 32 13 67 10 15 14 48 8 100 9 55;
6 71 5 64 2 40 0 32 3 92 1 59 4 69 13 68 14 34 12 71 8 28 9 94 7 82 10 1 11 58;
6 36 4 46 1 50 5 87 3 33 2 94 0 3 14 60 11 45 13 84 9 1 8 38 10 22 12 39 7 50;
1 53 0 34 5 56 6 97 3 95 4 32 2 28 14 48 7 54 12 98 8 84 9 77 10 46 13 65 11 94;
2 1 5 97 0 77 4 82 6 14 1 18 3 74 14 52 11 14 12 93 9 35 8 34 13 84 10 6 7 81;
1 62 0 86 2 57 6 80 5 37 3 94 4 77 7 72 9 26 11 41 10 7 8 56 13 98 14 67 12 47;
5 45 3 30 0 57 6 68 1 61 2 34 4 2 7 57 13 96 9 10 12 85 14 42 10 93 8 89 11 43;
6 49 4 53 1 51 2 4 0 17 5 21 3 31 10 45 13 45 9 63 11 21 8 4 7 23 14 90 12 1;
6 68 5 18 0 87 3 6 4 13 2 9 1 40 8 83 7 95 12 27 10 94 14 68 11 22 13 28 9 66;
2 80 6 14 0 67 5 15 1 14 3 97 4 23 8 45 10 1 11 5 14 87 7 34 12 12 9 98 13 35;
4 33 2 20 3 74 6 20 5 3 0 90 1 37 13 56 12 38 8 7 14 84 9 100 11 41 10 6 7 97;
6 47 4 63 3 1 0 28 2 99 1 41 5 45 14 60 13 2 7 25 8 59 9 39 10 76 11 89 12 5;
6 67 2 46 3 25 1 2 5 22 4 8 0 22 13 64 7 82 12 99 11 79 10 87 8 71 9 24 14 19;

- SWV09-hard(20×15)

5 8 3 73 0 69 2 38 6 6 4 62 1 78 9 79 8 59 13 77 11 22 10 80 12 58 14 49 7 48;
3 34 4 29 2 69 0 5 5 63 1 82 6 94 14 17 11 94 9 29 10 5 13 75 7 15 8 61 12 61;
1 52 2 30 0 25 6 17 3 46 4 86 5 3 14 70 11 34 9 23 10 68 13 76 8 53 12 71 7 9;
2 50 4 20 3 24 0 53 1 97 5 79 6 92 14 3 12 52 10 75 9 74 8 59 7 75 13 84 11 99;
2 15 0 61 3 47 4 38 6 49 5 21 1 6 11 8 8 71 14 83 13 24 12 18 9 33 7 70 10 100;
4 48 5 50 2 66 0 92 6 2 3 58 1 23 9 84 8 66 10 12 7 36 14 4 12 88 13 64 11 12;
3 29 0 25 6 44 5 87 2 42 1 44 4 86 8 28 10 86 9 74 14 77 13 59 12 94 7 58 11 16;
4 31 3 58 0 94 5 69 2 44 1 93 6 92 9 80 8 63 12 47 13 3 7 79 11 39 14 80 10 75;
1 69 2 27 0 76 5 19 6 86 3 16 4 31 12 33 9 69 13 19 10 43 14 9 11 37 7 35 8 24;
2 75 3 78 6 41 4 60 5 59 0 42 1 60 12 18 8 31 10 15 7 54 14 60 9 20 11 61 13 69;
4 89 6 20 1 27 5 78 3 2 2 21 0 55 13 79 11 77 10 99 9 70 12 30 7 97 8 41 14 98;
6 1 2 10 4 84 5 72 0 14 1 9 3 51 7 22 14 65 10 100 13 65 11 43 8 10 12 14 9 19;
5 50 2 13 3 49 6 75 1 42 0 81 4 89 9 100 14 54 13 37 10 7 11 38 8 25 12 78 7 79;
2 44 3 77 5 26 1 42 4 9 6 73 0 60 9 61 10 85 12 14 11 92 7 100 14 49 8 46 13 12;
2 72 0 53 1 43 5 65 6 59 4 87 3 13 8 71 12 25 9 71 10 89 11 2 7 76 14 21 13 12;
2 60 6 28 5 33 1 36 0 6 3 96 4 48 9 40 11 79 10 60 8 39 13 34 7 54 12 20 14 52;
5 82 2 12 3 11 4 61 1 21 0 21 6 34 12 86 14 53 8 7 9 4 7 95 10 62 13 54 11 82;
5 72 0 13 3 46 6 97 1 87 4 87 2 11 7 45 14 85 11 66 8 43 9 39 13 34 10 30 12 55;
1 39 5 19 0 19 4 73 6 63 3 30 2 69 9 36 7 13 10 96 12 27 13 59 14 76 11 62 8 14;
1 7 4 14 3 79 2 27 6 43 0 96 5 24 11 30 7 27 12 2 8 69 14 75 13 34 10 79 9 96;

- SWV10-hard(20×15)

3 8 2 73 1 79 0 95 6 69 4 9 5 5 8 85 9 52 11 43 14 32 7 91 10 24 13 89 12 38;
6 45 1 70 4 84 3 24 5 18 0 20 2 71 8 21 7 60 9 98 10 70 13 52 12 34 11 23 14 52;
6 16 4 68 1 85 0 39 5 40 2 98 3 61 10 77 7 60 11 73 9 66 14 84 8 16 13 43 12 88;
0 72 1 17 3 68 4 89 2 94 6 98 5 56 10 88 13 27 9 60 12 61 8 8 7 88 11 48 14 65;
6 78 2 24 5 28 0 73 4 21 1 69 3 52 14 32 8 83 11 48 10 29 13 48 12 92 9 43 7 82;
4 54 6 31 5 14 3 47 0 82 1 75 2 4 8 31 12 72 7 58 9 45 13 91 14 31 11 61 10 27;
4 88 1 28 5 92 6 62 3 93 0 14 2 65 7 33 9 44 8 31 14 32 11 72 13 47 12 61 10 34;
0 52 1 59 5 98 3 6 2 19 6 53 4 39 8 74 12 48 10 33 13 49 11 92 7 22 14 41 9 37;
0 2 6 85 3 34 2 51 4 97 5 95 1 73 14 61 9 28 12 73 8 21 11 85 7 75 13 42 10 7;
5 94 1 28 0 77 2 56 6 79 4 2 3 82 9 88 10 93 12 44 14 5 8 96 7 34 13 56 11 41;
2 15 5 88 6 18 3 14 1 82 0 58 4 33 13 19 10 42 9 36 14 57 12 85 7 3 11 62 8 36;
3 30 6 33 0 13 4 4 2 74 1 37 5 78 14 2 13 56 9 21 10 61 11 81 7 18 8 59 12 62;
5 40 1 75 6 45 0 41 3 97 2 65 4 92 7 11 12 44 8 40 9 100 11 91 14 66 13 53 10 27;
1 83 2 52 0 84 3 66 5 3 6 5 4 71 13 41 10 42 11 63 12 50 14 43 8 3 9 35 7 18;
4 44 0 26 1 59 6 81 2 84 5 81 3 91 13 41 7 42 11 53 8 63 14 89 9 15 10 64 12 40;
1 46 0 97 5 67 4 97 3 71 6 88 2 69 14 44 12 20 11 52 13 34 10 74 8 79 7 10 9 87;
3 71 6 13 4 100 2 67 1 57 5 24 0 36 7 88 14 79 8 21 9 86 12 60 11 28 10 14 13 3;
0 97 6 24 2 41 4 40 1 51 5 73 3 19 9 27 12 70 13 98 10 11 11 83 7 76 8 60 14 12;
5 88 3 48 1 33 4 96 6 10 0 49 2 52 10 38 13 49 7 31 12 94 14 23 9 7 11 5 8 4;
2 85 0 100 5 51 6 91 1 21 3 83 4 30 12 23 9 48 8 19 11 47 10 95 7 23 14 78 13 22;

• SWV11-hard(50×10)	• SWV12-hard(50×10)
0 92 4 47 3 56 2 91 1 49 5 39 9 63 7 12 6 1 8 37;	0 92 4 49 1 93 3 48 2 1 7 52 6 57 9 16 5 6 8 6;
0 86 2 100 1 75 3 92 4 90 5 11 7 85 8 54 9 100 6 38;	4 82 3 25 2 69 1 86 0 54 6 15 5 31 9 5 7 6 8 18;
1 4 4 94 3 44 2 40 0 92 8 53 6 40 9 5 5 68 7 27;	0 31 1 26 3 46 2 49 4 48 8 71 7 82 5 47 9 93 6 91;
4 87 0 48 1 59 2 92 3 35 6 99 7 46 9 27 8 83 5 91;	0 34 4 37 1 82 3 25 2 43 6 11 9 71 5 55 7 34 8 77;
0 83 1 78 4 76 3 64 2 44 8 12 9 91 6 31 7 98 5 63;	4 22 0 91 3 54 2 49 1 97 9 2 7 46 5 98 6 27 8 89;
3 49 0 15 1 100 4 18 2 24 6 92 9 65 5 26 7 29 8 24;	2 46 3 70 1 3 0 44 4 24 9 65 6 60 5 94 8 58 7 22;
0 28 3 53 4 84 2 47 1 85 7 100 5 34 6 35 8 90 9 88;	3 53 0 99 1 80 2 74 4 29 6 72 7 54 5 98 8 60 9 69;
2 61 4 71 3 54 1 34 0 13 9 47 8 2 6 97 7 27 5 97;	3 96 1 87 0 36 2 57 4 7 8 36 9 26 5 94 6 47 7 70;
0 85 2 75 1 33 4 72 3 49 7 23 5 12 8 90 6 87 9 42;	3 5 2 47 1 59 0 57 4 28 9 24 8 79 6 19 5 44 7 35;
2 24 3 20 1 65 4 33 0 75 9 47 6 84 8 44 7 74 5 29;	0 96 1 4 3 60 2 45 4 39 7 97 5 2 9 81 6 89 8 91;
2 48 3 27 4 1 0 23 1 66 6 35 7 46 9 29 5 63 8 44;	2 23 4 74 3 98 0 24 1 75 9 57 8 93 6 74 5 10 7 44;
2 79 0 1 4 61 3 46 1 59 7 10 8 88 9 19 6 50 5 34;	3 36 4 5 2 36 0 49 1 90 8 62 5 74 9 4 6 85 7 53;
0 16 4 31 3 77 2 3 1 25 8 88 7 97 9 49 6 79 5 22;	2 44 1 47 3 75 4 81 0 30 7 42 8 100 9 81 6 29 5 31;
1 40 0 39 4 15 2 93 3 48 6 63 9 74 8 46 7 91 5 51;	1 2 0 18 3 88 2 27 4 5 5 36 7 30 6 51 8 51 9 31;
4 48 0 93 2 8 3 50 1 5 6 48 7 46 9 35 5 88 8 97;	1 21 0 57 3 100 2 100 4 59 8 77 7 21 5 98 6 38 9 84;
3 70 1 8 2 65 0 32 4 84 8 9 6 43 7 10 5 72 9 60;	4 97 2 72 1 70 3 99 0 42 6 94 5 59 9 90 8 78 7 13;
0 21 2 28 1 26 3 91 4 58 9 90 6 43 8 61 5 39 7 93;	3 16 2 19 1 70 0 7 4 74 6 7 5 50 9 74 8 46 7 88;
1 50 2 60 0 51 4 90 3 93 7 20 9 33 8 27 6 12 5 89;	3 45 4 91 2 28 0 52 1 12 5 45 6 7 7 15 9 22 8 31;
1 21 3 3 2 47 4 34 0 53 9 67 8 8 5 68 7 1 6 71;	3 56 2 3 1 8 4 25 0 90 8 99 6 22 9 65 7 51 5 31;
3 57 4 26 2 36 0 48 1 11 9 44 7 25 5 30 8 92 6 57;	0 23 3 28 1 49 2 5 4 17 7 40 9 30 5 62 8 65 6 84;
1 20 0 20 4 6 3 74 2 48 9 77 8 15 5 80 7 27 6 10;	2 88 0 86 4 8 1 41 3 12 6 67 9 77 5 94 7 80 8 11;
3 71 1 40 0 86 2 23 4 29 7 99 8 56 6 100 9 77 5 28;	4 81 3 42 0 19 2 100 1 10 5 23 9 71 8 18 6 93 7 36;
4 83 0 61 3 27 1 86 2 99 7 31 5 60 8 40 9 84 6 26;	4 74 2 73 3 63 1 9 0 51 8 39 7 7 6 96 5 81 9 22;
4 68 1 94 3 46 2 60 0 33 7 46 5 86 9 63 6 70 8 89;	1 1 3 44 0 66 4 19 2 65 7 10 6 23 8 26 9 76 5 77;
4 33 1 13 2 91 3 27 0 38 8 82 7 31 6 23 9 27 5 87;	1 54 2 18 4 99 0 79 3 22 5 2 6 42 8 54 7 90 9 28;
4 58 3 30 0 24 2 12 1 38 8 2 9 37 5 59 6 77 7 36;	3 16 4 1 1 28 0 54 2 97 5 71 6 53 8 32 7 26 9 28;
2 62 1 47 4 5 3 39 0 75 7 60 9 65 8 61 6 77 5 31;	0 82 3 5 2 18 4 71 1 50 5 41 7 62 9 89 6 93 8 54;
4 100 0 21 1 53 3 74 2 3 8 34 6 6 7 91 9 60 5 28;	2 63 3 59 0 42 1 74 4 32 5 50 6 21 7 29 8 83 9 6[illegible];
1 8 0 3 2 88 3 54 4 18 9 4 6 34 5 54 8 59 7 42;	4 29 2 76 1 6 3 44 0 4 9 81 5 29 7 95 8 66 6 89;
3 33 4 72 0 83 2 17 1 23 6 24 8 60 9 96 7 78 5 70;	3 55 4 84 1 36 0 42 2 64 5 81 8 85 6 76 7 4 9 16;
4 63 2 36 3 70 0 97 1 99 6 71 9 92 5 41 8 73 7 97;	4 100 0 46 1 69 3 41 2 3 6 18 5 41 7 94 8 97 9 30;
2 28 1 37 4 24 0 30 3 55 8 38 5 9 9 77 7 17 6 51;	3 34 4 35 2 18 1 58 0 98 9 78 8 17 5 53 6 85 7 86;
3 15 0 46 2 14 4 18 1 99 9 48 6 41 5 10 7 47 8 80;	4 68 2 89 1 99 0 3 3 92 5 10 6 52 7 30 8 66 9 69;
4 89 3 78 2 51 1 63 0 29 7 70 9 7 5 14 8 84 6 32;	0 21 3 65 4 19 2 14 1 76 9 84 6 45 5 24 8 54 7 73;
4 26 1 69 2 92 3 15 0 23 8 42 6 95 5 47 9 83 7 56;	4 47 0 68 2 87 3 92 1 96 6 29 5 90 8 29 7 39 9 100;
1 38 2 44 3 47 4 23 0 10 9 63 7 65 6 21 5 70 8 56;	2 35 0 60 4 61 1 61 3 72 9 57 8 94 5 77 7 1 6 53;
3 42 4 85 1 29 0 35 2 66 9 46 8 25 5 90 7 85 6 75;	3 85 2 38 0 79 4 43 1 71 6 44 5 87 8 61 7 51 9 37;
3 99 0 46 4 74 2 96 1 48 5 52 6 13 7 88 8 4 9 30;	1 100 2 33 3 94 0 59 4 25 5 88 9 50 6 19 8 4 7 66;
1 15 3 80 4 47 2 25 0 8 9 61 7 70 8 23 6 93 5 5;	2 8 0 85 1 80 4 75 3 1 7 17 9 32 6 60 5 30 8 57;
0 90 2 51 3 66 4 5 1 86 5 59 6 97 9 28 7 85 8 9;	4 25 2 98 1 94 3 49 0 34 9 37 7 80 6 50 8 25 5 72;
0 59 1 50 4 40 3 23 2 93 7 61 9 96 8 63 6 34 5 14;	3 51 4 49 1 53 2 7 0 73 6 96 7 19 9 41 5 55 8 42;
1 62 2 72 4 30 0 21 3 15 5 77 6 13 7 2 8 22 9 22;	0 57 1 86 2 1 4 61 3 66 6 28 5 56 7 68 8 21 9 65;
2 20 4 14 3 85 1 4 0 2 9 33 7 90 5 48 8 90 6 62;	2 98 1 100 0 47 4 28 3 4 7 34 9 55 5 32 6 72 8 66;
0 49 3 49 4 46 1 89 2 64 9 72 8 6 5 83 6 13 7 66;	4 2 0 74 2 20 1 39 3 63 5 88 9 3 7 22 6 8 8 73;
4 74 1 55 2 73 0 25 3 16 7 19 9 38 6 22 5 26 8 63;	2 44 0 1 3 52 1 43 4 4 6 36 9 75 8 58 5 61 7 38;
3 13 2 96 1 8 0 15 4 97 6 95 7 2 5 66 8 57 9 46;	2 21 4 6 3 32 1 74 0 57 5 72 8 10 9 34 6 91 7 94;
4 73 1 97 3 39 0 22 2 90 9 64 6 65 8 31 5 98 7 85;	4 26 0 59 3 53 1 45 2 23 5 55 8 12 7 34 6 98 9 43;
3 43 2 67 0 38 1 77 4 11 7 61 5 7 9 95 8 97 6 69;	2 4 1 53 4 57 3 95 0 6 6 30 8 1 7 92 9 20 5 86;
0 35 2 68 1 5 3 46 4 4 7 51 6 44 5 58 9 69 8 98;	1 98 2 77 3 65 4 51 0 85 7 23 6 79 5 30 8 41 9 17;
2 68 1 81 0 2 3 4 4 59 9 53 8 69 5 69 6 14 7 21;	4 58 2 43 3 14 0 74 1 64 7 37 8 78 6 33 9 42 5 80;

• SWV13-hard(50×10)	• SWV14-hard(50×10)
4 68 1 39 2 79 0 72 3 65 5 82 7 33 6 82 8 66 9 55;	4 69 0 37 3 64 1 1 2 65 9 34 5 67 8 43 7 72 6 79;
2 14 3 45 0 18 4 72 1 27 7 57 6 90 8 19 9 19 5 50;	1 11 0 7 3 68 4 43 2 52 6 29 9 71 7 81 8 12 5 36;
4 25 1 77 0 64 3 18 2 19 8 27 6 97 9 81 7 65 5 11;	4 90 3 29 1 1 2 1 0 14 8 38 5 13 9 21 7 41 6 97;
3 70 0 29 2 31 1 39 4 62 8 12 9 2 5 91 7 98 6 91;	1 46 0 26 4 83 2 36 3 20 9 4 8 23 7 65 5 56 6 42;
2 90 4 51 3 38 1 27 0 29 6 67 8 95 9 60 7 86 5 64;	4 46 0 39 2 92 3 53 1 62 9 68 7 65 8 74 6 87 5 16;
4 90 0 55 3 69 1 76 2 97 7 94 5 57 8 65 9 80 6 24;	4 13 1 41 3 43 2 67 0 75 6 5 9 94 5 95 7 28 8 85;
1 23 4 13 0 90 3 24 2 41 8 69 7 8 5 81 6 94 9 76;	1 1 2 99 4 36 3 86 0 65 8 32 5 17 7 71 6 15 9 61;
3 19 1 37 0 16 4 4 2 68 6 45 8 79 9 4 7 30 5 33;	2 18 4 63 3 15 0 59 1 33 7 95 5 63 6 85 8 34 9 3;
2 36 0 76 3 97 4 71 1 19 9 87 6 97 8 64 5 84 7 43;	4 13 2 25 0 82 3 23 1 26 7 22 9 35 8 16 6 24 5 41;
2 20 1 77 0 71 3 73 4 47 7 88 5 100 9 16 8 69 6 77;	3 1 1 7 0 21 2 73 4 39 6 32 7 77 5 29 8 89 9 21;
3 55 4 96 0 8 2 61 1 40 8 46 7 29 9 71 5 89 6 59;	1 53 3 27 4 55 0 16 2 64 5 78 9 32 8 60 7 20 6 20;
2 21 0 18 3 37 4 97 1 59 7 79 6 2 5 80 8 85 9 59;	1 71 2 54 3 21 0 20 4 23 9 40 5 99 7 61 6 94 8 71;
4 19 1 83 2 1 0 95 3 48 9 37 7 59 5 56 8 57 6 81;	2 76 4 72 3 91 0 75 1 7 6 53 8 32 7 71 5 63 9 53;
0 8 1 60 4 91 3 85 2 27 9 39 5 31 6 62 7 94 8 12;	2 12 1 3 4 35 0 64 3 30 5 94 8 67 7 31 6 79 9 14;
4 2 3 10 0 17 1 38 2 96 6 21 9 81 8 64 5 73 7 46;	4 63 1 28 3 87 0 89 2 52 8 2 9 21 7 92 6 44 5 37;
2 46 1 4 4 25 3 41 0 11 5 96 9 56 6 10 7 25 8 32;	0 79 1 65 4 35 3 78 2 17 8 90 5 54 9 91 7 57 6 23;
0 21 1 77 4 22 2 72 3 53 9 28 7 23 5 2 8 52 6 83;	3 20 1 93 4 61 0 76 2 23 5 10 8 34 7 20 9 87 6 77;
3 9 4 37 0 2 2 74 1 15 8 26 5 83 6 90 7 51 9 80;	0 37 2 17 1 92 4 30 3 59 5 47 8 7 7 45 6 13 9 60;
3 6 1 7 0 57 2 4 4 56 7 11 5 57 8 12 6 94 9 29;	4 90 3 74 0 46 2 36 1 2 6 9 5 83 8 90 7 88 9 39;
1 40 2 93 3 65 4 66 0 96 9 5 7 52 8 85 5 93 6 94;	3 83 0 85 2 20 4 88 1 94 6 14 5 16 7 62 9 53 8 9;
1 38 2 19 4 22 0 73 3 7 5 63 8 28 6 23 9 11 7 84;	0 4 4 16 2 64 1 60 3 79 5 37 6 49 7 67 9 95 8 5;
1 96 4 10 0 29 3 59 2 94 5 26 7 22 8 52 6 37 9 50;	3 32 0 86 1 5 4 66 2 77 7 15 5 68 9 40 8 1 6 4;
1 38 3 31 2 76 0 8 4 8 6 50 5 95 8 5 9 25 7 62;	0 2 1 48 4 23 3 25 2 58 9 55 7 14 8 21 6 85 5 27;
0 15 2 84 4 100 3 76 1 66 7 56 5 95 8 94 6 56 9 85;	1 71 4 92 3 99 2 56 0 81 7 79 6 66 9 42 8 47 5 43;
3 73 2 38 1 84 0 42 4 37 5 16 7 24 9 59 6 60 8 23;	1 77 4 85 3 72 2 19 0 71 5 34 7 9 9 14 6 62 8 58;
3 43 1 79 0 80 2 44 4 65 5 81 7 7 8 93 6 55 9 34;	4 38 0 3 2 61 3 98 1 76 5 14 9 56 8 26 7 43 6 44;
2 8 4 2 0 12 3 55 1 60 9 91 6 6 5 83 8 31 7 91;	1 68 4 54 0 62 2 93 3 22 6 57 7 79 9 19 5 77 8 43;
0 8 4 46 3 47 2 57 1 47 9 55 8 74 7 98 6 54 5 51;	2 62 1 96 4 56 0 68 3 24 5 41 6 19 7 2 8 73 9 50;
2 56 4 90 1 41 0 35 3 62 7 4 5 15 9 89 6 73 8 66;	2 86 0 53 3 3 1 89 4 37 7 100 5 59 9 23 6 19 8 35;
0 2 4 39 3 44 1 68 2 54 7 7 8 76 9 29 5 90 6 53;	3 90 4 94 0 21 2 78 1 85 5 94 6 90 8 28 9 92 7 56;
2 34 0 94 3 1 1 23 4 45 8 83 7 84 5 49 6 67 9 49;	4 85 2 97 0 8 3 27 1 86 9 26 7 5 8 96 5 68 6 57;
4 4 2 70 1 19 0 19 3 92 5 70 7 33 9 50 8 82 6 48;	0 58 3 4 4 49 2 1 1 79 8 10 6 44 9 87 5 16 7 13;
4 64 2 76 0 70 3 83 1 91 7 98 8 37 5 3 9 75 6 92;	3 85 0 24 4 23 1 41 2 59 8 20 6 52 5 58 9 75 7 77;
3 96 1 17 0 20 4 13 2 28 7 21 9 65 5 87 6 54 8 98;	0 47 1 89 2 68 4 88 3 17 6 48 8 84 9 100 5 92 7 47;
0 68 4 40 3 98 2 90 1 38 7 45 8 21 5 9 9 3 6 47;	1 30 0 1 3 61 4 20 2 73 8 78 7 41 9 52 5 43 6 74;
0 58 4 19 2 16 3 74 1 32 9 32 5 58 6 93 7 1 8 80;	0 11 4 58 3 66 2 67 1 18 8 42 7 88 9 49 5 62 6 71;
0 32 2 99 1 95 3 2 4 8 9 55 6 32 8 26 5 6 7 68;	4 5 2 51 3 67 1 20 0 11 7 37 6 42 8 25 9 57 5 1;
3 7 4 45 2 19 0 97 1 56 7 22 9 72 8 98 5 59 6 20;	0 58 4 83 2 9 3 68 1 21 6 28 9 77 5 19 7 32 8 66;
2 97 4 98 3 43 0 28 1 23 5 3 8 75 9 43 7 58 6 71;	3 85 2 58 0 65 1 80 4 50 7 79 5 43 8 29 9 9 6 18;
3 31 0 88 2 88 1 82 4 65 5 53 9 15 7 68 6 60 8 99;	3 74 2 29 0 11 1 23 4 34 7 84 8 57 5 77 6 83 9 82;
4 4 0 100 2 95 1 11 3 28 5 80 7 25 9 87 6 25 8 9;	2 6 4 67 0 97 3 66 1 21 8 90 9 46 6 12 5 17 7 96;
0 75 3 10 4 59 2 80 1 60 5 75 8 87 6 33 9 10 7 31;	4 34 1 5 2 13 0 100 3 12 8 63 7 59 5 75 6 91 9 89;
0 54 3 6 4 7 1 72 2 49 7 72 8 64 6 32 9 86 5 69;	1 30 2 66 0 33 3 70 4 16 6 80 5 58 8 8 7 86 9 66;
4 15 3 19 1 18 0 84 2 96 9 71 8 64 6 38 5 58 7 62;	3 55 0 46 2 1 1 77 4 19 7 85 9 32 6 59 5 37 8 69;
1 32 4 80 2 83 3 83 0 50 5 81 7 82 9 33 8 10 6 55;	2 3 0 16 1 48 4 8 3 51 7 72 6 19 8 58 9 59 5 94;
0 65 4 95 3 84 2 64 1 18 9 27 6 70 7 74 5 87 8 68;	3 30 4 23 1 92 0 18 2 19 9 32 6 57 5 30 7 64 8 27;
1 50 2 49 0 96 3 1 4 89 8 42 5 88 9 91 6 64 7 3;	2 18 0 72 4 92 1 6 3 67 8 100 6 32 9 14 5 51 7 55;
3 44 0 91 1 5 2 100 4 77 6 20 5 13 7 25 9 71 8 71;	4 48 0 87 1 96 2 58 3 83 8 77 5 26 7 77 9 72 6 86;
0 86 4 91 1 19 2 69 3 71 5 13 8 87 6 98 9 43 7 13;	1 80 4 5 0 50 3 65 2 85 7 88 5 47 6 33 8 50 9 75;
4 8 0 60 3 31 2 93 1 8 9 1 7 19 6 8 5 85 8 24;	1 78 0 96 4 80 3 5 2 99 9 58 5 38 7 29 8 69 6 44;

• SWV15-hard(50×10)	• SWV16-easy(50×10)
2 93 4 40 0 1 3 77 1 77 5 16 9 74 8 11 6 51 7 92;	1 53 3 46 5 71 8 29 0 47 2 12 7 57 4 79 6 91 9 30;
0 92 4 80 1 76 3 59 2 70 5 86 9 17 6 78 7 30 8 93;	2 96 6 94 8 98 0 55 3 10 1 95 5 95 7 37 9 82 4 2;
1 44 2 92 3 96 4 77 0 53 9 10 7 49 5 84 8 59 6 11;	6 43 3 93 8 30 2 41 0 23 1 60 7 14 4 15 5 42 9 56;
1 60 2 19 3 76 0 73 4 85 7 13 8 93 5 68 9 50 6 78;	0 45 6 85 2 59 7 76 1 93 9 62 4 33 8 46 5 33 3 35;
2 20 0 24 3 41 1 2 4 4 9 44 7 79 8 81 5 16 6 39;	2 45 3 36 8 11 6 96 7 96 1 8 0 75 5 6 4 13 9 2;
3 41 2 35 1 32 4 18 0 15 8 98 6 29 5 19 7 14 9 26;	9 51 7 75 0 4 3 13 5 12 1 4 2 38 6 30 4 42 8 28;
1 59 0 45 4 53 3 44 2 98 5 84 6 23 7 45 8 39 9 89;	9 58 4 33 6 77 2 11 3 37 8 64 5 94 7 89 1 96 0 93;
1 30 4 51 3 25 0 51 2 84 6 60 5 45 7 89 8 25 9 97;	6 37 3 67 0 88 9 92 8 19 4 27 7 46 1 58 2 60 5 55;
0 47 3 18 2 40 4 62 1 58 5 36 7 93 8 77 9 90 6 15;	4 60 2 88 0 23 5 69 8 60 1 32 7 4 6 56 9 25 3 14;
3 33 1 68 0 41 4 72 2 20 6 69 7 47 5 22 9 47 8 22;	2 98 5 56 1 68 6 63 7 61 3 78 8 45 0 62 4 31 9 70;
2 28 1 100 4 20 0 35 3 26 5 24 9 41 6 42 7 100 8 32;	7 66 8 80 0 18 3 97 9 47 5 38 1 26 2 8 6 90 4 90;
0 65 2 12 4 53 3 93 1 40 8 18 7 23 5 60 6 89 9 53;	0 16 7 6 4 53 6 86 5 81 8 49 3 90 2 57 1 34 9 36;
0 58 1 60 4 97 3 31 2 50 9 85 5 64 7 38 6 85 8 35;	2 69 8 65 5 20 4 15 1 61 3 71 6 71 9 58 0 24 7 71;
3 64 0 58 1 49 2 45 4 9 8 49 6 22 5 99 9 15 7 7;	4 84 5 20 9 58 0 53 8 98 2 75 7 46 3 81 1 71 6 46;
0 10 4 85 3 72 2 37 1 77 5 70 7 45 9 8 6 83 8 57;	5 6 6 58 7 90 1 54 9 73 0 92 4 39 3 23 2 100 8 18;
4 93 0 87 1 87 2 18 3 4 8 78 5 67 9 20 6 17 7 35;	2 32 5 58 6 97 1 49 3 61 0 69 8 2 4 3 9 32 7 16;
4 72 0 56 3 57 2 15 1 45 6 41 5 40 9 85 8 32 7 81;	0 78 7 14 4 98 3 26 8 25 9 45 6 12 2 98 1 99 5 69;
0 36 3 63 4 79 2 32 1 5 6 25 7 86 9 91 5 21 8 35;	2 50 1 95 4 82 9 25 0 68 8 83 5 36 7 78 3 35 6 27;
2 83 4 29 0 9 1 38 3 73 7 50 9 99 5 18 8 29 6 41;	6 29 7 20 8 55 4 14 2 66 5 52 0 75 9 63 1 93 3 64;
0 100 3 29 2 60 4 63 1 64 8 71 6 35 5 26 9 9 7 22;	1 11 0 18 9 42 4 81 7 2 2 39 3 83 6 11 5 38 8 52;
1 81 0 60 3 62 4 48 2 68 7 28 5 69 8 92 6 79 9 10;	4 11 8 99 9 2 7 10 3 91 5 83 6 61 0 21 2 69 1 8;
0 40 4 80 1 41 2 10 3 68 8 28 9 51 7 33 6 82 5 25;	9 11 7 65 1 14 2 85 3 5 8 5 5 11 4 47 6 67 0 41;
4 30 2 12 0 35 3 17 1 70 9 29 7 18 8 93 6 94 5 37;	9 60 7 9 8 16 2 4 5 34 6 2 4 30 1 32 0 51 3 51;
1 36 2 41 3 27 4 36 0 78 7 64 6 88 5 25 9 92 8 66;	9 31 2 41 1 13 6 28 5 97 3 8 7 42 4 95 8 46 0 93;
2 65 3 27 4 74 0 32 1 40 5 88 8 73 6 92 7 83 9 42;	4 1 6 91 8 49 3 75 1 19 7 100 0 58 2 14 5 34 9 82;
0 48 1 85 2 92 4 95 3 61 8 72 9 76 5 58 7 11 6 89;	3 28 5 68 9 30 7 68 1 10 6 20 8 47 4 51 0 44 2 32;
3 84 2 50 0 70 4 24 1 42 9 55 5 100 6 70 7 4 8 68;	9 86 3 9 1 80 0 89 5 93 4 12 8 13 7 10 6 18 2 4;
0 95 4 41 2 11 3 98 1 85 5 64 6 8 7 26 8 6 9 6;	0 22 5 12 8 95 4 24 3 30 1 81 2 21 7 28 9 100 6 27;
0 84 2 49 1 17 3 69 4 55 8 75 6 45 9 38 7 59 5 28;	1 87 0 68 2 64 3 33 7 59 5 95 6 1 9 14 8 82 4 43;
2 48 0 29 4 1 1 64 3 41 5 23 7 64 9 31 6 56 8 12;	2 14 6 98 0 86 1 85 8 85 5 12 4 99 7 8 3 21 9 7;
2 81 4 25 3 33 0 22 1 50 5 74 9 56 8 33 7 35 6 83;	5 47 9 90 0 88 1 52 8 43 4 62 7 33 3 51 6 97 2 22;
1 62 4 25 0 21 2 20 3 8 6 36 9 9 5 91 8 90 7 49;	2 59 7 26 4 76 0 26 3 71 8 59 1 73 9 70 5 57 6 10;
1 43 0 16 2 91 3 96 4 24 5 11 9 91 7 41 8 35 6 66;	6 92 2 10 9 45 0 11 1 53 3 35 8 76 4 83 7 55 5 79;
1 91 2 20 4 44 0 42 3 87 9 57 6 15 5 38 8 42 7 89;	9 96 4 3 3 92 7 67 6 60 8 35 5 70 0 52 2 39 1 94;
0 33 3 95 4 68 2 22 1 80 7 53 8 13 9 70 5 22 6 69;	4 65 0 17 9 26 7 46 5 81 1 42 2 64 6 46 3 96 8 59;
0 15 3 47 1 24 2 31 4 41 8 14 9 28 7 59 5 52 6 39;	9 6 3 21 8 46 0 82 2 74 5 56 7 94 6 83 4 63 1 21;
2 95 0 42 4 5 1 57 3 67 6 30 9 21 8 70 5 9 7 20;	6 89 5 23 8 78 2 33 9 4 7 97 3 60 1 29 0 79 4 93;
2 54 0 15 1 20 3 64 4 83 9 40 7 6 5 89 6 91 8 48;	0 46 1 46 4 20 7 91 2 76 9 83 3 14 6 61 5 84 8 76;
0 22 4 27 1 77 3 25 2 16 8 72 9 61 6 75 7 4 5 19;	7 82 8 43 6 76 1 36 0 27 9 93 5 71 4 81 2 45 3 62;
3 68 1 82 2 16 0 83 4 2 7 10 8 88 5 41 9 21 6 66;	7 51 9 27 5 12 6 52 4 85 8 66 0 100 3 44 2 82 1 36;
1 64 0 76 2 85 3 71 4 97 5 97 7 8 6 40 8 70 9 35;	3 75 7 13 6 63 1 78 4 1 8 60 2 24 5 10 9 56 0 3;
0 94 1 45 2 94 4 84 3 44 8 41 5 30 7 47 6 19 9 22;	5 48 4 32 2 82 0 1 1 2 7 35 3 16 9 67 8 74 6 39;
2 23 1 10 0 82 3 93 4 90 8 67 7 9 9 18 5 22 6 87;	7 24 0 8 8 96 3 59 2 41 4 23 1 37 9 4 5 69 6 27;
0 75 2 27 4 97 3 9 1 57 9 14 5 50 7 31 8 62 6 23;	1 23 9 3 2 85 6 93 5 18 7 47 0 96 8 6 4 60 3 3;
1 42 3 41 2 35 0 75 4 18 9 65 7 38 6 38 8 51 5 56;	6 99 2 14 9 16 3 81 8 89 1 53 7 86 4 39 5 3 0 87;
4 72 1 63 0 33 2 27 3 41 5 52 7 42 9 10 6 11 8 71;	5 67 8 53 0 77 4 69 2 55 3 78 6 95 1 76 7 2 9 71;
2 91 1 89 0 44 4 91 3 26 6 49 5 22 8 31 9 69 7 5;	1 5 6 89 0 37 3 88 7 20 9 4 4 77 8 27 5 31 2 47;
3 42 1 34 0 4 4 34 2 16 6 86 7 25 8 99 5 67 9 25;	1 66 2 55 4 15 7 35 3 76 9 91 6 35 5 37 8 54 0 33;
4 34 1 95 0 26 3 81 2 9 7 96 8 79 9 68 5 76 6 10;	3 79 5 2 6 17 1 65 7 27 8 53 4 52 9 35 0 23 2 59;
3 19 1 47 4 13 2 98 0 32 7 12 9 45 6 52 8 49 5 34;	9 100 0 55 5 14 2 86 4 69 3 87 8 46 1 3 6 89 7 100;

- SWV17-easy(50×10)

7 9 2 57 9 62 5 34 6 83 0 33 1 80 4 46 3 21 8 89;
9 82 1 35 8 37 5 26 6 21 3 78 7 64 4 33 2 40 0 21;
7 14 5 49 3 48 9 34 4 52 1 16 2 78 0 24 8 58 6 43;
2 94 3 86 8 41 5 27 7 29 6 53 9 5 0 36 4 98 1 37;
7 55 1 87 8 51 5 29 9 93 3 51 0 54 6 85 2 20 4 29;
2 88 1 98 3 67 8 41 6 23 9 70 7 26 4 28 5 17 0 87;
2 78 0 18 4 43 3 86 9 78 6 43 7 62 8 42 1 44 5 9;
9 37 4 89 3 26 6 59 0 89 5 90 1 91 8 28 7 37 2 51;
3 82 2 31 1 98 5 25 0 16 7 23 9 92 4 89 6 32 8 12;
6 66 1 58 5 14 3 42 0 62 8 66 4 46 7 88 2 89 9 97;
8 94 9 11 6 3 1 86 2 4 5 19 7 93 4 43 0 78 3 11;
5 22 1 87 9 61 2 2 3 15 6 37 7 81 0 17 8 31 4 73;
6 28 0 86 3 54 2 68 4 63 1 33 8 22 5 35 9 84 7 15;
6 18 1 2 2 23 8 49 7 82 9 8 4 73 5 31 3 20 0 1;
7 49 5 8 2 36 8 31 6 47 3 90 0 7 9 6 1 44 4 51;
4 43 1 95 0 18 9 99 7 98 3 26 8 99 5 90 2 24 6 91;
1 49 6 69 3 73 9 52 0 10 7 41 8 42 5 96 4 85 2 76;
0 5 1 69 3 38 7 35 5 23 2 40 8 17 4 33 6 99 9 82;
3 42 1 93 4 90 6 88 2 70 8 11 9 51 7 76 5 40 0 94;
5 88 9 44 0 63 7 92 1 4 4 91 6 92 8 53 3 52 2 38;
5 83 3 75 1 44 2 79 7 63 6 32 0 10 4 2 9 6 8 56;
7 71 0 23 5 93 3 44 6 36 1 27 2 96 4 23 9 35 8 21;
5 42 2 43 6 37 9 98 0 55 3 35 4 45 1 8 8 5 7 100;
0 40 8 34 2 7 9 17 5 30 4 98 7 34 6 23 1 37 3 58;
9 87 2 39 3 23 8 48 6 83 7 50 5 9 1 49 0 37 4 42;
6 60 5 3 2 60 7 40 0 54 1 68 4 49 8 50 9 22 3 34;
5 22 1 55 2 32 0 83 8 38 4 22 6 29 7 23 9 59 3 90;
9 51 2 27 6 81 8 87 0 79 7 1 3 14 5 73 4 25 1 11;
6 88 1 46 5 16 2 62 9 95 7 63 4 78 0 9 3 68 8 37;
4 77 2 13 8 96 3 61 0 21 7 39 5 12 6 19 9 73 1 86;
7 91 5 14 3 37 0 17 9 49 4 27 1 68 2 60 6 42 8 15;
9 13 4 25 6 62 0 4 1 31 8 76 5 3 7 8 3 26 2 95;
7 45 5 50 1 14 0 69 9 43 4 1 6 73 8 35 3 1 2 61;
4 57 1 1 0 71 8 1 6 96 2 92 7 85 5 42 3 12 9 38;
7 49 5 31 8 79 6 83 1 40 4 65 3 34 2 32 9 97 0 25;
9 24 5 40 4 81 3 10 6 59 8 83 2 66 1 28 7 33 0 31;
5 33 4 39 3 50 1 96 7 62 2 72 8 42 6 86 9 66 0 80;
3 88 7 47 0 35 4 69 1 79 9 61 2 25 8 56 5 68 6 96;
9 23 6 95 0 42 1 84 8 57 4 42 2 2 5 79 3 29 7 90;
9 96 8 21 4 17 7 12 1 25 2 9 6 7 5 26 0 81 3 51;
1 63 7 16 6 40 2 22 9 48 5 87 0 15 8 24 3 37 4 55;
7 95 0 60 3 62 2 7 9 2 8 81 5 83 4 64 1 68 6 66;
3 24 7 60 6 35 2 77 1 85 8 57 9 29 5 59 4 53 0 14;
1 24 6 30 0 9 3 89 8 72 4 77 2 7 5 23 9 73 7 35;
0 66 8 12 1 9 5 50 2 14 9 76 4 90 3 43 7 18 6 63;
3 97 1 29 0 59 4 64 9 17 2 77 5 60 7 16 6 61 8 40;
9 5 4 22 2 3 8 63 5 1 7 23 0 1 3 61 1 92 6 19;
6 91 8 74 1 88 5 2 7 61 4 39 0 35 2 23 9 84 3 27;
8 87 5 58 7 44 1 6 6 22 3 57 9 78 4 19 2 71 0 6;
4 6 1 94 0 45 2 54 9 67 7 90 5 19 8 72 6 70 3 58;

- SWV18-easy(50×10)

7 35 6 23 2 92 4 5 5 40 1 90 3 30 9 35 8 8 0 86;
2 60 3 97 8 21 9 70 7 82 0 12 4 3 5 45 1 75 6 69;
7 96 2 38 0 61 1 55 4 31 5 18 9 79 3 4 6 12 8 29;
4 83 7 82 8 97 1 43 0 95 6 92 2 18 3 29 5 4 9 67;
3 46 9 80 8 66 2 38 4 95 1 40 7 89 0 32 6 64 5 1;
6 57 4 80 8 68 7 27 0 90 5 45 3 98 9 59 1 6 2 94;
5 50 0 91 2 97 9 63 7 52 3 46 4 4 8 96 1 18 6 100;
7 23 6 13 3 25 8 83 2 76 9 41 1 88 0 31 5 44 4 13;
2 20 3 90 9 20 4 42 8 72 5 46 1 27 0 81 6 40 7 34;
7 80 5 97 0 42 2 49 9 10 1 10 3 71 4 71 6 14 8 98;
2 79 3 29 0 96 7 66 1 58 8 31 4 47 5 76 6 59 9 88;
8 93 6 [illegible] 1 7 3 27 5 66 7 23 0 60 4 97 2 66 9 55;
9 12 8 59 4 77 5 79 0 26 7 58 2 98 6 38 3 31 1 28;
6 8 9 48 4 4 1 87 3 38 2 28 8 10 0 19 7 82 5 83;
5 6 9 13 2 86 6 19 3 26 7 79 0 55 1 85 8 33 4 30;
3 37 8 26 7 29 6 71 9 43 5 17 0 45 2 28 1 58 4 15;
7 15 3 37 6 21 5 47 2 90 0 37 9 33 1 42 4 7 8 62;
8 49 4 46 1 28 7 18 6 41 2 57 0 75 3 21 9 3 5 32;
6 98 1 30 8 24 4 91 9 73 7 25 5 49 0 40 2 9 3 4;
6 33 3 94 1 21 2 90 9 86 7 85 5 29 0 17 4 94 8 90;
6 3 4 85 1 66 7 61 8 57 3 84 2 5 9 40 0 54 5 70;
7 81 1 98 2 45 0 18 6 65 9 1 4 98 3 30 8 84 5 82;
6 40 7 77 3 72 1 97 5 39 4 21 0 59 8 42 9 90 2 26;
5 57 3 33 1 14 4 64 6 23 8 78 2 54 0 51 9 100 7 96;
5 61 1 55 6 73 2 87 4 35 3 41 7 96 0 32 8 91 9 60;
9 19 5 90 8 91 0 45 3 66 2 84 1 61 7 3 6 84 4 100;
2 33 9 72 6 27 8 14 3 59 0 39 7 20 5 29 4 54 1 88;
4 45 0 18 3 73 2 26 8 55 6 22 7 27 1 15 9 43 5 77;
2 57 9 16 1 71 8 25 7 50 3 41 6 58 5 71 4 9 0 32;
8 18 9 32 0 42 3 73 1 56 7 53 6 3 5 66 4 15 2 44;
6 69 7 11 1 2 8 40 4 70 9 90 3 38 2 31 5 55 0 50;
9 100 8 14 0 55 2 3 5 12 4 79 1 68 3 83 6 89 7 78;
4 26 5 44 8 39 1 84 7 61 9 98 3 38 2 2 6 27 0 18;
3 98 2 10 9 99 8 50 0 20 6 12 4 7 1 57 7 87 5 89;
0 64 8 63 7 98 5 31 1 30 6 62 3 11 4 89 9 31 2 34;
3 26 6 43 4 69 7 27 8 92 2 51 1 10 5 29 9 21 0 37;
8 21 5 98 0 64 6 38 2 23 1 13 7 89 9 89 4 21 3 27;
4 39 7 32 1 67 0 33 5 16 2 43 6 62 3 42 9 70 8 90;
7 73 9 45 3 37 0 45 2 61 6 25 5 15 4 5 8 58 1 98;
7 94 0 17 6 15 5 81 9 64 3 62 1 2 8 16 2 35 4 40;
5 32 6 37 9 11 0 25 1 37 8 21 2 76 7 52 4 56 3 87;
3 23 2 40 1 6 7 31 6 25 9 98 8 29 4 4 5 25 0 33;
8 96 9 30 1 95 3 2 6 3 2 22 0 62 4 30 7 1 5 99;
9 54 5 3 0 78 2 43 6 90 7 88 4 1 8 97 1 30 3 96;
5 29 6 60 3 80 1 94 2 67 0 42 8 17 9 27 7 75 4 86;
1 17 5 62 2 25 7 80 6 62 9 19 8 81 3 73 0 57 4 90;
9 31 3 54 5 28 1 19 4 4 2 34 8 64 6 16 7 60 0 27;
9 95 7 1 2 43 3 6 4 7 8 66 1 45 5 13 0 80 6 1;
3 20 7 82 0 87 1 65 6 64 8 61 2 21 5 32 9 16 4 37;
0 49 3 54 2 31 8 69 1 21 5 2 6 73 9 35 4 66 7 82;

- SWV19 easy(50×10)

7 74 1 27 5 66 3 89 6 58 0 11 8 77 9 17 2 70 4 97;
5 10 0 11 2 38 3 60 1 30 7 35 6 94 9 52 4 2 8 20;
7 17 0 65 6 93 8 62 9 91 3 2 1 51 2 4 3 19 4 10;
4 87 3 3 9 81 0 17 6 44 2 82 7 16 5 13 8 100 1 85;
9 18 6 33 7 35 0 78 2 68 3 68 8 3 5 2 4 53 1 25;
2 36 8 41 6 60 9 43 0 66 5 34 3 21 7 11 1 5 4 55;
9 52 4 99 6 62 0 30 1 24 8 73 7 19 3 23 2 15 5 2;
4 85 9 21 3 27 7 53 0 86 1 36 6 35 5 99 8 30 2 43;
6 43 5 31 9 99 2 12 0 6 7 79 3 81 1 18 8 73 4 55;
4 90 6 100 1 15 0 40 7 96 9 25 5 43 8 23 2 31 3 7;
5 61 4 88 6 10 3 48 0 100 2 62 1 83 8 20 7 42 9 19;
9 35 7 41 6 16 3 58 0 86 2 69 5 58 1 93 4 47 8 77;
2 61 0 40 4 99 1 51 7 16 6 39 3 43 9 37 8 38 5 9;
4 15 8 38 2 84 5 98 6 17 1 91 7 91 9 23 3 48 0 98;
3 26 2 42 8 55 4 24 0 43 1 83 9 27 7 38 6 37 5 58;
5 21 8 78 6 97 0 77 9 82 4 26 3 22 1 90 7 57 2 31;
4 3 9 44 3 90 1 64 5 52 8 35 7 18 2 45 0 4 6 14;
8 60 6 59 3 67 2 85 0 43 7 93 5 44 4 22 1 58 9 38;
4 77 8 11 2 74 6 99 0 100 1 45 9 14 3 26 7 98 5 77;
8 38 9 57 7 12 5 64 1 80 6 81 4 70 3 13 2 41 0 65;
9 36 4 22 8 39 0 76 1 78 2 27 5 55 3 10 6 5 7 71;
7 70 9 81 1 60 5 85 3 63 6 97 2 61 8 44 0 5 4 35;
9 38 0 94 2 46 5 20 8 87 1 41 4 41 3 40 7 99 6 48;
7 30 6 9 5 13 2 79 8 81 0 25 9 93 4 85 3 78 1 76;
4 6 8 58 6 51 7 48 2 68 3 34 5 78 9 59 1 98 0 36;
4 90 6 56 7 97 9 37 0 38 1 47 2 56 3 8 5 37 8 7;
0 66 8 15 1 39 5 89 7 3 9 54 3 24 2 14 6 99 4 73;
3 12 9 37 4 79 8 95 0 50 1 74 6 1 5 55 7 98 2 49;
8 99 9 79 3 99 2 87 0 80 4 13 5 99 6 13 1 54 7 61;
1 51 9 21 3 32 6 20 0 80 7 58 2 91 5 84 8 62 4 91;
1 11 8 38 2 14 9 12 3 39 5 34 0 37 6 94 4 10 7 2;
6 76 9 86 3 40 4 30 2 97 0 59 8 100 7 9 5 55 1 86;
3 33 1 49 0 94 2 17 6 17 8 70 5 17 7 42 4 26 9 24;
4 75 1 20 9 93 2 58 3 51 0 94 6 24 7 70 8 51 5 82;
8 59 1 9 3 59 5 62 9 79 7 53 6 48 4 98 2 76 0 71;
6 90 2 35 5 89 0 59 9 28 7 51 4 69 3 36 1 32 8 27;
5 10 6 85 4 97 1 3 0 79 9 86 3 10 7 80 2 37 8 39;
7 60 0 27 5 69 8 58 6 67 2 36 9 31 3 69 1 16 4 22;
2 27 5 16 6 15 4 40 8 16 1 92 9 60 7 43 3 2 0 7;
1 79 7 99 0 27 9 56 5 29 6 17 8 67 4 34 3 86 2 61;
6 57 7 100 4 73 9 17 8 3 3 64 2 99 0 71 5 27 1 90;
2 80 5 23 4 54 6 39 9 77 3 65 7 59 0 7 1 63 8 32;
4 98 6 17 8 44 5 1 3 10 7 56 2 95 9 80 0 99 1 64;
8 60 7 74 3 60 6 30 0 81 5 25 4 89 9 19 2 59 1 21;
1 67 0 42 8 93 2 47 5 34 7 11 6 100 9 15 4 99 3 2;
9 35 3 61 5 3 8 83 7 87 4 66 0 96 2 55 1 41 6 61;
8 22 5 25 7 29 3 70 6 93 1 19 0 49 9 62 2 19 4 73;
8 11 4 93 5 97 1 28 2 14 0 75 7 41 3 40 9 62 6 66;
7 76 6 61 8 64 3 90 0 20 2 43 9 50 1 13 5 4 4 47;
3 38 4 11 0 30 5 37 7 57 9 64 1 68 8 42 2 19 6 79;

- SWV20-easy(50×10)

8 100 7 30 4 42 9 11 2 31 1 71 5 41 0 1 3 55 6 94;
4 81 6 20 3 96 7 39 8 29 0 90 9 61 2 64 1 86 5 47;
5 80 0 56 1 88 7 19 2 68 8 95 3 44 4 22 9 60 6 80;
4 86 6 70 0 88 2 15 7 50 1 51 9 88 3 25 8 89 5 33;
0 48 1 57 4 86 8 60 3 78 5 4 9 60 7 40 2 11 6 25;
6 23 7 9 1 90 0 51 2 52 9 14 5 30 4 1 8 25 3 83;
1 30 4 75 5 76 9 100 7 54 2 41 6 50 8 75 0 1 3 28;
2 46 3 78 1 37 7 12 6 56 4 50 8 66 5 39 0 8 9 72;
1 24 6 90 0 32 3 6 2 99 9 22 8 12 4 63 7 81 5 52;
6 62 3 9 8 59 0 66 4 41 1 32 5 29 7 79 9 84 2 4;
9 57 5 99 6 2 3 17 0 51 7 10 4 14 1 64 2 99 8 27;
7 81 0 67 9 83 2 30 5 25 6 87 1 29 3 7 8 93 4 1;
5 65 8 53 9 18 4 28 7 74 0 60 6 77 2 22 1 5 3 98;
1 97 5 37 0 71 7 49 6 51 3 17 4 38 9 67 8 28 2 31;
0 20 8 94 3 39 6 73 9 63 4 8 2 57 1 27 7 26 5 12;
8 77 1 68 9 20 7 100 4 1 5 77 6 17 3 35 2 65 0 86;
8 68 6 62 4 79 7 84 1 60 3 56 0 10 9 86 5 60 2 30;
4 71 2 74 6 6 1 56 3 69 0 8 8 50 9 78 5 4 7 89;
8 29 5 5 1 59 3 96 0 16 4 91 2 48 7 53 6 21 9 82;
2 19 9 96 0 73 1 39 5 54 8 50 7 60 3 50 4 65 6 78;
7 68 4 15 2 26 3 26 0 13 9 13 5 96 8 70 6 27 1 93;
6 41 8 78 4 66 7 9 1 31 2 92 0 3 3 78 5 41 9 53;
5 9 0 64 2 15 6 73 4 12 1 43 8 89 7 69 3 32 9 22;
5 93 6 19 3 74 8 81 0 72 2 91 9 19 1 26 4 53 7 7;
3 48 2 29 5 51 8 72 7 35 6 32 1 38 0 98 4 58 9 54;
0 91 9 23 4 41 6 53 2 53 7 27 1 62 3 68 8 84 5 49;
4 4 1 4 0 66 7 90 9 78 2 29 5 2 6 86 3 23 8 46;
3 78 5 61 2 97 7 68 8 92 0 15 4 12 6 77 1 12 9 22;
0 100 7 89 6 71 2 70 8 89 4 72 5 78 3 23 9 37 1 2;
0 91 3 74 2 36 4 72 6 62 1 80 9 20 7 77 5 47 8 80;
1 44 0 67 4 66 8 99 6 59 5 5 7 15 2 38 3 40 9 19;
1 69 9 35 3 86 0 7 2 35 5 32 6 66 4 89 8 63 7 52;
3 3 4 68 1 66 7 27 6 41 5 2 9 77 0 45 2 40 8 39;
4 66 3 42 7 79 0 55 6 98 9 44 5 6 8 73 1 55 2 1;
3 80 8 18 9 94 2 27 5 42 4 17 7 74 0 65 6 6 1 27;
2 73 4 70 5 51 0 84 8 29 9 95 1 97 7 28 3 68 6 89;
9 85 6 56 5 54 3 76 2 50 0 43 1 8 7 93 4 17 8 65;
1 1 3 17 2 61 5 38 4 71 7 18 0 40 9 94 6 41 8 74;
3 30 8 22 6 39 9 56 5 3 7 61 4 74 2 21 0 93 1 1;
0 17 8 8 9 20 5 38 3 85 7 5 2 63 1 18 4 89 6 88;
8 87 5 44 0 42 1 34 9 11 7 13 3 71 4 88 6 32 2 12;
2 39 1 73 6 43 0 48 9 77 8 48 5 23 7 66 3 94 4 68;
1 98 7 19 3 69 6 5 8 85 9 19 0 30 2 43 5 87 4 70;
2 45 1 60 4 30 9 71 5 35 0 75 3 75 6 41 8 67 7 37;
3 63 7 39 2 16 9 69 1 46 5 20 6 57 4 52 0 66 8 40;
2 7 7 73 6 17 1 21 0 24 8 2 5 68 4 22 9 36 3 60;
1 20 4 17 8 12 9 29 5 28 0 7 3 38 6 57 7 22 2 75;
5 53 4 7 7 5 8 27 9 38 2 100 6 48 0 53 1 11 3 18;
1 49 7 47 4 81 8 9 0 20 2 63 3 15 6 1 9 10 5 5;
4 49 6 27 7 17 5 64 2 30 8 56 0 42 3 97 9 82 1 34;

附录 1.6 YN 类(4 个子问题)

• YN1(20×20)

17 13 2 26 11 35 4 45 12 29 13 21 7 40 0 45 3 16 15 10 18 49 10 43 14 25 8 25 1 40 6 16 19 43 5 48 9 36 16 11;
8 21 6 22 14 15 5 28 10 10 2 46 11 19 19 13 13 18 18 14 3 11 4 21 16 30 1 29 0 16 15 41 17 40 12 38 7 28 9 39;
4 39 3 28 8 32 17 46 0 35 14 14 1 44 10 20 13 12 6 23 18 22 9 15 11 35 7 27 16 26 5 27 15 23 2 27 12 31 19 31;
4 31 10 24 3 34 6 44 18 43 12 32 2 35 15 34 19 21 7 46 13 15 5 10 9 24 14 37 17 38 1 41 8 34 0 32 16 11 11 36;
19 45 1 23 5 34 9 23 7 41 16 10 11 40 12 46 14 27 8 13 4 20 2 40 15 28 13 44 17 34 18 21 10 27 0 12 6 37 3 30;
13 48 2 34 3 22 7 14 12 22 14 10 8 45 19 38 6 32 16 38 11 16 4 20 0 12 5 40 9 33 17 35 1 32 10 15 15 31 18 49;
9 19 5 33 18 32 16 37 12 28 3 16 2 40 10 37 4 10 11 20 1 17 17 48 6 44 13 29 14 44 15 48 8 21 0 31 7 36 19 43;
9 20 6 43 1 13 5 22 2 33 7 28 16 39 12 16 13 34 17 20 10 47 18 43 19 44 8 29 15 22 4 14 11 28 14 44 0 33 3 28;
7 14 12 40 8 19 0 49 13 11 10 13 9 47 18 22 2 27 17 26 3 47 5 37 6 19 15 43 14 41 1 34 11 21 4 30 19 32 16 45;
16 32 7 22 15 30 6 18 18 41 19 34 9 22 11 11 17 29 10 37 4 30 2 25 1 27 0 31 14 16 13 20 3 26 12 14 5 24 8 43;
18 22 17 22 12 30 15 31 13 15 4 13 16 47 19 18 6 33 3 30 7 46 2 48 11 42 0 18 1 16 8 25 10 43 5 21 9 27 14 14;
5 48 1 39 2 21 18 18 13 20 0 28 15 20 8 36 6 24 9 35 7 22 19 36 3 39 14 34 4 49 17 36 11 38 10 46 12 44 16 13;
14 26 1 32 2 11 15 10 9 41 13 10 6 26 19 25 12 13 11 35 5 22 0 11 7 24 17 33 8 11 10 34 16 11 3 22 4 12 18 17;
16 39 10 21 17 43 14 28 3 49 15 31 18 46 13 29 6 31 11 40 7 24 1 47 9 15 2 26 8 40 12 46 5 18 19 16 4 14 0 21;
11 41 19 26 16 14 3 47 0 49 5 16 17 31 9 43 15 20 10 25 14 10 13 49 8 32 6 36 7 19 4 23 2 20 18 15 12 34 1 33;
11 37 5 48 10 31 7 42 2 24 1 13 9 30 15 24 0 19 13 34 19 35 8 42 3 10 14 40 4 39 6 42 12 38 16 12 18 27 17 40;
14 19 1 27 8 39 12 41 5 45 11 40 10 46 6 48 7 37 3 30 17 31 4 16 18 29 15 44 0 41 16 55 13 47 9 21 2 10 19 48;
18 38 0 27 13 32 9 30 7 17 14 21 1 14 4 37 17 15 16 31 5 27 10 25 15 41 11 48 3 48 6 36 2 30 12 45 8 26 19 17;
1 17 10 40 9 16 5 36 4 34 16 47 19 14 0 24 18 10 6 14 13 14 3 30 12 23 2 37 17 11 11 23 8 40 15 15 14 10 7 46;
14 37 10 28 13 13 0 28 2 18 1 43 16 46 8 39 3 30 12 15 11 38 17 38 18 45 19 44 9 16 15 29 5 33 6 20 7 35 4 34;

• YN2(20×20)

17 15 2 28 11 10 4 46 12 19 13 13 7 18 0 14 3 11 15 21 18 30 10 29 14 16 8 41 1 40 6 38 19 28 5 39 9 39 16 28;
8 32 6 46 14 35 5 14 10 44 2 20 11 12 19 23 13 22 18 15 3 35 4 27 16 26 1 27 0 23 15 27 17 31 12 31 7 31 9 24;
4 34 3 44 8 43 17 32 0 35 14 34 1 21 10 40 13 15 6 10 18 24 9 37 11 38 7 41 16 34 5 32 15 11 2 36 12 45 19 23;
4 34 10 23 3 41 6 10 18 40 12 46 2 27 15 13 19 20 7 40 13 28 5 44 9 34 14 21 17 27 1 12 8 37 0 30 16 48 11 34;
19 22 1 14 5 22 9 10 7 45 16 38 11 32 12 38 14 16 8 20 4 12 2 40 15 33 13 35 17 32 18 15 10 31 0 49 6 19 3 33;
13 32 2 37 3 28 7 16 12 40 14 37 8 10 19 20 6 17 16 48 11 44 4 29 0 44 5 48 9 21 17 31 1 36 10 43 15 20 18 43;
9 13 5 22 18 33 16 28 12 39 3 16 2 34 10 20 4 47 11 43 1 44 17 29 6 22 13 14 14 28 15 44 8 33 0 28 7 11 19 40;
9 19 6 49 1 11 5 13 2 47 7 22 16 27 12 26 13 47 17 37 10 19 18 43 19 41 8 34 15 21 4 30 11 32 14 45 0 32 3 22;
7 30 12 18 8 41 0 34 13 22 10 11 9 29 18 37 2 30 17 25 3 27 5 31 6 16 15 20 14 26 1 14 11 24 4 43 19 22 16 22;
16 30 7 31 15 15 6 13 18 47 19 18 9 33 11 30 17 46 4 48 10 42 2 18 1 16 0 25 14 43 13 21 3 27 12 14 5 48 8 39;
18 21 17 18 12 20 15 28 13 20 4 36 16 24 19 35 7 22 3 36 6 39 10 34 11 49 0 36 1 38 8 46 9 44 5 13 2 26 14 32;
9 11 1 10 2 41 11 10 13 26 0 26 12 13 10 35 6 22 5 11 7 24 19 33 3 11 14 34 17 11 4 22 18 12 8 17 15 39 16 24;
1 43 15 28 2 49 14 34 4 46 12 29 18 31 19 40 13 24 11 47 5 15 0 26 7 40 17 46 8 18 10 16 16 14 5 21 9 41 6 26;
16 14 6 47 17 49 10 16 3 31 12 43 4 20 8 25 14 10 18 49 7 32 0 36 9 19 2 23 15 20 5 15 13 34 19 33 11 37 1 48;
4 31 11 42 7 24 6 13 0 30 14 24 17 19 19 34 16 35 10 42 15 10 13 40 2 39 8 12 5 38 9 12 1 27 18 40 12 19 3 27;
6 39 5 41 13 45 15 40 2 46 9 48 7 37 0 30 1 31 12 16 19 29 14 44 3 41 8 35 10 47 11 21 4 10 16 48 18 38 17 27;
16 32 1 30 8 17 18 21 0 14 17 57 10 15 12 31 7 27 3 25 5 41 4 48 13 48 6 36 2 30 15 45 11 26 9 17 14 17 19 40;
18 16 17 36 4 34 2 47 10 14 15 24 1 10 3 14 7 14 12 30 5 23 9 37 8 11 14 23 11 40 6 15 16 10 0 46 13 37 19 28;
17 13 13 28 11 18 16 43 7 46 8 39 3 30 5 15 4 38 2 38 14 45 0 44 10 16 6 29 12 33 1 20 19 35 15 34 9 16 18 40;
17 14 2 30 0 27 15 47 18 43 3 17 14 13 6 43 7 45 12 32 13 13 16 48 1 10 4 14 10 42 9 38 5 43 19 22 11 43 8 23;

• YN3(20×20)

13 47 16 21 17 27 8 46 1 27 14 39 19 24 4 34 7 27 3 36 6 11 5 32 0 13 9 40 2 40 15 20 18 45 10 23 12 36 11 31;
1 40 11 20 12 27 6 32 16 26 13 36 10 37 7 26 3 22 4 44 18 18 2 11 17 15 9 27 15 39 5 25 8 16 14 13 0 49 19 25;
9 40 8 11 14 47 2 35 13 41 7 37 1 37 18 28 6 42 3 23 10 41 5 33 17 25 0 19 19 15 16 42 12 37 11 34 4 10 15 41;
2 28 4 18 11 42 5 26 13 27 6 24 12 41 0 25 1 27 7 40 17 40 14 49 10 33 3 30 15 34 16 17 8 49 9 21 18 35 19 42;
7 26 9 27 4 25 3 42 19 28 15 22 17 34 0 15 6 46 1 34 12 47 2 16 16 34 10 31 14 24 5 43 13 45 11 47 8 18 18 15;
4 30 8 48 1 46 15 13 9 20 7 31 14 20 2 20 16 34 19 38 18 12 17 11 11 47 5 19 0 35 13 17 10 23 12 11 3 22 6 11;
3 27 2 11 5 17 0 43 1 25 15 24 18 36 8 12 9 21 13 44 10 17 17 41 16 34 11 14 12 45 7 45 14 27 6 47 4 47 19 11;
5 27 4 41 17 44 16 16 11 42 10 29 3 23 2 15 0 22 13 28 7 16 14 39 9 21 12 15 18 32 15 36 1 29 8 18 6 39 19 33;
4 44 19 38 11 24 17 21 13 34 15 11 10 16 8 43 16 41 7 45 3 37 9 10 6 36 18 31 2 17 14 28 12 43 0 22 1 25 5 15;
7 40 15 23 4 37 2 12 8 28 12 19 10 30 17 40 13 20 18 11 5 23 16 46 3 40 1 37 14 17 0 16 11 31 6 15 9 10 19 22;
5 10 1 37 15 22 2 28 6 10 9 21 19 38 16 35 7 34 0 13 14 33 11 16 4 26 3 20 17 10 18 37 13 21 8 31 10 27 12 23;
16 32 6 32 7 20 1 14 0 11 19 27 3 21 18 32 10 33 13 13 17 36 8 25 4 32 5 41 15 44 2 32 14 12 9 32 12 10 11 28;
7 28 9 33 11 35 17 44 4 43 16 35 12 31 2 14 6 48 8 40 15 28 0 31 3 22 5 30 13 27 10 24 18 47 14 38 1 46 19 22;
12 33 6 33 14 38 9 15 10 16 13 24 1 30 8 18 7 46 2 30 17 37 11 24 5 13 3 14 18 11 16 38 0 31 4 24 19 42 15 30;
10 15 16 12 6 43 18 27 0 24 9 20 3 41 2 22 12 41 11 30 5 26 4 24 7 45 13 46 14 22 15 11 8 20 1 42 19 11 17 49;
4 14 19 30 17 15 7 17 8 34 2 48 3 45 14 16 12 23 16 29 13 28 6 28 18 24 10 21 5 37 1 38 11 31 0 29 9 42 15 22;
15 41 17 19 5 37 7 36 8 47 12 49 11 29 6 18 9 33 10 30 0 49 16 37 3 11 2 46 14 36 18 35 13 45 1 31 4 33 19 18;
9 42 4 11 15 28 18 48 6 22 8 15 1 37 11 36 3 26 19 21 2 48 16 17 12 30 10 27 13 35 17 20 0 18 7 14 14 20 5 41;
19 35 17 19 16 20 15 36 1 15 3 46 4 13 8 12 18 10 5 37 2 10 13 44 10 30 11 20 14 42 6 35 0 26 9 29 7 21 12 42;
17 33 3 11 7 42 16 45 9 29 0 27 5 15 13 37 2 32 11 25 14 21 8 49 19 34 1 31 15 35 6 32 4 20 18 30 10 24 12 29;

• YN4(20×20)

16 34 17 38 0 21 6 15 15 42 8 17 7 41 18 10 10 26 11 24 1 31 19 25 14 31 13 33 4 35 9 30 3 16 12 16 5 30 2 13;
5 41 11 33 6 15 16 38 0 40 14 38 3 37 1 20 13 22 4 34 7 16 17 39 9 15 2 19 10 36 12 39 18 26 8 19 15 39 19 34;
17 34 1 12 16 10 7 47 13 28 15 27 0 19 6 34 19 33 12 40 9 37 14 24 8 15 10 34 2 44 3 37 18 22 11 31 4 39 5 26;
5 48 7 46 16 47 10 45 14 15 8 25 0 31 3 24 12 35 18 15 2 48 13 19 11 10 1 48 17 16 15 28 4 18 6 17 9 44 19 41;
12 47 3 23 9 48 16 45 14 39 6 42 8 32 15 11 13 16 5 14 11 19 1 46 19 10 10 17 7 41 2 47 17 32 4 17 0 21 18 17;
18 14 16 20 1 18 12 14 13 10 6 16 5 24 4 18 0 21 11 18 15 42 19 13 3 23 14 40 9 48 8 12 2 24 10 23 7 45 17 30;
0 27 12 15 4 26 13 19 17 14 5 49 7 16 18 25 16 16 8 20 9 36 2 21 14 30 3 36 1 17 15 22 6 43 11 32 10 23 19 17;
0 32 16 15 17 12 7 46 3 37 18 43 11 40 13 43 9 48 4 36 15 24 8 25 1 33 14 32 5 26 6 37 12 24 10 24 2 15 19 22;
10 34 6 33 15 25 8 46 0 20 18 33 4 19 13 45 2 47 1 32 3 12 11 29 16 29 5 46 12 17 7 48 14 39 17 40 19 41 9 37;
13 26 3 47 5 44 6 49 1 22 17 12 10 28 19 36 9 27 4 25 14 48 7 11 16 49 12 24 11 48 2 19 0 47 18 49 8 46 15 36;
13 23 18 48 14 15 0 42 3 36 8 15 6 32 10 18 1 15 15 23 11 45 2 13 17 21 12 32 7 44 5 25 19 34 16 22 9 11 4 43;
17 37 7 49 15 45 2 28 9 15 8 35 12 29 13 44 1 26 4 25 5 30 3 39 0 15 14 28 18 23 6 42 11 33 16 45 10 10 19 20;
0 10 6 37 3 15 13 13 10 11 2 49 1 28 14 28 15 13 8 29 12 21 16 32 11 21 4 48 5 11 17 26 9 33 18 22 7 21 19 49;
18 38 0 41 4 30 13 43 6 11 2 43 14 27 3 26 9 30 15 19 16 36 1 31 17 47 5 41 10 34 8 40 12 32 7 13 11 18 19 27;
6 24 5 30 7 10 10 35 8 28 16 43 19 12 9 14 15 15 3 15 2 35 18 43 0 38 4 16 1 29 17 40 14 49 13 38 12 16 11 30;
3 48 6 35 13 43 2 37 17 18 5 27 9 27 7 41 1 22 15 28 16 18 10 37 18 48 4 10 8 14 11 15 14 43 0 48 12 12 19 49;
0 13 13 38 7 34 6 42 1 36 5 45 18 24 8 35 14 26 19 30 12 47 16 21 11 47 4 40 10 43 3 16 15 10 2 12 9 39 17 22;
16 30 13 47 19 49 8 20 4 40 3 46 17 21 14 33 6 44 7 23 9 24 0 48 10 43 15 41 2 32 5 29 11 36 1 38 12 47 18 12;
13 10 5 36 12 18 16 48 0 27 14 43 10 46 6 27 7 46 19 35 11 31 2 18 8 24 3 23 17 29 18 14 9 19 1 40 15 38 4 13;
9 45 16 44 0 43 17 31 14 35 13 17 12 42 3 14 18 37 10 39 6 48 7 38 15 26 4 19 2 28 11 35 1 42 5 24 8 44 19 38;

附录 1.7 TD 或 TA 类(80 个子问题)

在此问题不直接给出数据，而是给出可产生该 80 个算例的数据的 C 语言程序。该程序可选择机器编号是由 0 还是 1 开始，对于每个算例所产生的数据，第 1 行包括工件数和机器数，接下来的每行(按工件号由 1 到 n 排)数据表示每个工件加工先后顺序对应的机器号及相应加工时间。所测试的计算机/系统/编译器有如下几种：DEC 5000/ Ultrix 4.2/ cc; SUN 10/ Solaris 2/ cc; IBM 6000/ AIX 3.x/ xlc; Atari ST/ TOS/ pure c; IBM-PC/ DOS 6.2/ MS-C; IBM-PC/ Linux/ gcc。必须注意，程序要求'long'和'double'型数据为 64 位(bit)。

程序取条件 VERIFY == 1 and FIRMACIND == 1 时，生成验证文件 ta01，产生的 4 工件 4 机器的数据如下：

```
4 4
3 54 1 34 4 61 2 2
4 9 1 15 2 89 3 70
1 38 2 19 3 28 4 87
1 95 3 34 2 7 4 29
```

产生 TD 类 JSP 算例的 C 程序如下：

```
#define ANSI_C 0            /* 0:K&R 函数形式转换 */
#define VERIFY 0            /* 1:产生验证文件 */
#define FIRMACIND 0         /* 0,1:起始机器编号 */
#include <stdio.h>
#include <math.h>
struct problem {
    long rand_time;         /* 工件的随机种子 */
    long rand_mach;         /* 机器的随机种子 */
    short num_jobs;         /* 工件数 */
    short num_mach;         /* 机器数 */
};

#if VERIFY == 1
struct problem S[] = {
    {0, 0, 0, 0},
    {1166510396, 164000672, 4, 4},
```

```
  {0, 0, 0, 0}};

#else /* VERIFY */
struct problem S[] = {
{0, 0, 0, 0},

/* 15 工件 15 机器 */
{840612802, 398197754, 15, 15},
{1314640371, 386720536, 15, 15},
{1227221349, 316176388, 15, 15},
{342269428, 1806358582, 15, 15},
{1603221416, 1501949241, 15, 15},
{1357584978, 1734077082, 15, 15},
{44531661, 1374316395, 15, 15},
{302545136, 2092186050, 15, 15},
{1153780144, 1393392374, 15, 15},
{73896786, 1544979948, 15, 15},

/* 20 工件 15 机器 */
{533484900, 317419073, 20, 15},
{1894307698, 1474268163, 20, 15},
{874340513, 509669280, 20, 15},
{1124986343, 1209573668, 20, 15},
{1463788335, 529048107, 20, 15},
{1056908795, 25321885, 20, 15},
{195672285, 1717580117, 20, 15},
{961965583, 1353003786, 20, 15},
{1610169733, 1734469503, 20, 15},
{532794656, 998486810, 20, 15},

/* 20 工件 20 机器 */
{1035939303, 773961798, 20, 20},
{5997802, 1872541150, 20, 20},
{1357503601, 722225039, 20, 20},
{806159563, 1166962073, 20, 20},
{1902815253, 1879990068, 20, 20},
{1503184031, 1850351876, 20, 20},
```

```
{1032645967, 99711329, 20, 20},
{229894219, 1158117804, 20, 20},
{823349822, 108033225, 20, 20},
{1297900341, 489486403, 20, 20},

/* 30工件15机器 */
{98640593, 1981283465, 30, 15},
{1839268120, 248890888, 30, 15},
{573875290, 2081512253, 30, 15},
{1670898570, 788294565, 30, 15},
{1118914567, 1074319202, 30, 15},
{178750207, 294279708, 30, 15},
{1549372605, 596993084, 30, 15},
{798174738, 151685779, 30, 15},
{553410952, 1329272528, 30, 15},
{1661531649, 1173386294, 30, 15},

/* 30工件20机器 */
{1841414609, 1357882888, 30, 20},
{2116959593, 1546338557, 30, 20},
{796392706, 1230864158, 30, 20},
{532496463, 254174057, 30, 20},
{2020525633, 978943053, 30, 20},
{524444252, 185526083, 30, 20},
{1569394691, 487269855, 30, 20},
{1460267840, 1631446539, 30, 20},
{198324822, 1937476577, 30, 20},
{38071822, 1541983579, 30, 20},

/* 50工件15机器 */
{17271, 718939, 50, 15},
{660481279, 149650254, 50, 15},
{352229765, 949737911, 50, 15},
{1197518780, 166840558, 50, 15},
{1376020303, 483922052, 50, 15},
{2106639239, 955932362, 50, 15},
{1765352082, 1209982549, 50, 15},
{1105092880, 1349003108, 50, 15},
```

```
{907248070, 919544535, 50, 15},
{2011630757, 1845447001, 50, 15},

/* 50工件20机器 */
{8493988, 2738939, 50, 20},
{1991925010, 709517751, 50, 20},
{342093237, 786960785, 50, 20},
{1634043183, 973178279, 50, 20},
{341706507, 286513148, 50, 20},
{320167954, 1411193018, 50, 20},
{1089696753, 298068750, 50, 20},
{433032965, 1589656152, 50, 20},
{615974477, 331205412, 50, 20},
{236150141, 592292984, 50, 20},

/* 100工件20机器 */
{302034063, 1203569070, 100, 20},
{1437643198, 1692025209, 100, 20},
{1792475497, 1039908559, 100, 20},
{1647273132, 1012841433, 100, 20},
{696480901, 1689682358, 100, 20},
{1785569423, 1092647459, 100, 20},
{117806902, 739059626, 100, 20},
{1639154709, 1319962509, 100, 20},
{2007423389, 749368241, 100, 20},
{682761130, 262763021, 100, 20},

{0, 0, 0, 0}};
#endif /* VERIFY */

/* 在上下界内均匀产生随机数 */
#if ANSI_C == 1
short unif (long *seed, short low, short high)
#else
short unif (seed, low, high)
long *seed; short low, high;
#endif
```

```
{
    static long m = 2147483647, a = 16807, b = 127773, c = 2836;
    double value_0_1;
    long k = *seed / b;
    *seed = a * (*seed % b) - k * c;
    if(*seed < 0) *seed = *seed + m;
    value_0_1 = *seed / (double) m;
    return low + floor(value_0_1 * (high-low + 1));
}

/* 最多支持100个工件20台机器,对于更大规模的问题需要扩大数组大小 */
short d[101][21];               /* 加工时间 */
short M[101][21];               /* 加工机器号 */
#if ANSI_C == 1
void generate_job_shop(short p)    /* 根据S[p]填入将d和M */
#else
void generate_job_shop(p)
short p;
#endif
{
    short i, j;
    long time_seed = S[p].rand_time;
    long machine_seed = S[p].rand_mach;
    for(i = 0; i < S[p].num_jobs; ++i)                /* 确定一个随机加工时间 */
        for (j = 0; j < S[p].num_mach; ++j)           /* 对所有的操作 */
            d[i][j] = unif(&time_seed, 1, 99);        /* 各操作的加工时间不超过99 */

    for(i = 0; i < S[p].num_jobs; ++i)                /* 确定一台机器 */
        for (j = 0; j < S[p].num_mach; ++j)           /* 对所有的操作 */
            M[i][j] = j;                              /* 分配机器j */

    for(i = 0; i < S[p].num_jobs; ++i) {              /* 对所有工件 */
        for (j = 0; j < S[p].num_mach; ++j) {         /* 对所有机器 */
            int k = unif(&machine_seed, j, S[p].num_mach-1);    /* 得到随机指示 */

            short t = M[i][j];                        /* 交换位置j和k */
            M[i][j] = M[i][k];
```

```
          M[i][k] = t;
       }
   }
}

#if ANSI_C == 1
void write_problem(short p)              /* 写问题 */
#else
void write_problem(p)
short p;
#endif
{
   short i, j;
   FILE *f = NULL;
   char name[5];
   sprintf(name,"ta%02d", p);              /* 构造文件名 */
   if(! (f = fopen(name,"w"))) {           /* 打开欲写的文件 */
      fprintf(stderr,"file %s error\n", name);
      return;
   }
   fprintf(f,"%d %d\n", S[p].num_jobs, S[p].num_mach); /* 写第 1 行 */
   for(i = 0; i < S[p].num_jobs; ++i) {
      for(j = 0; j < S[p].num_mach; ++j) {
         fprintf(f,"%2d %2d ", M[i][j]+FIRMACIND, d[i][j]);
                                                     /* 写机器号和加工时间 */
      }
      fprintf(f,"\n");                     /* 写完一个工件后换行 */
   }
   fclose(f);                              /* 关闭文件 */
}

int main()
{
   short i = 1;
   while(S[i].rand_time) {                 /* i 由 1 直到写完所有问题 */
      generate_job_shop(i);                /* 生成问题 i */
      write_problem(i);                    /* 写问题 i */
      ++i;                                 /* 转入下一个问题 */
   }
   return 0;
}
```

附录 2 典型 Flow Shop 调度问题

附录 2 给出本书 4.2 节介绍的若干典型 Flow Shop 调度问题的加工数据等信息，以便读者进行相关研究。对于每个算例，第 k 行数据（各行数据用分号隔开）表示工件 k 按加工先后顺序（由机器 0 至机器 $m-1$ 或由机器 1 至机器 m）排列的各操作在相应机器上的加工时间。譬如，Car1 算例的第 1 行数据表示工件 1 先在机器 0 上加工 375 单位时间，然后在机器 1 上加工 12 单位时间，然后在机器 2 上加工 142 单位时间，依次类推，最后在机器 4 上加工 412 单位时间。

附录 2.1 Car 类（8 个子问题）

• Car1（11×5）

375 12 142 245 412;
632 452 758 278 398;
12 876 124 534 765;
460 542 523 120 499;
528 101 789 124 999;
796 245 632 375 123;
532 230 543 896 452;
14 124 214 543 785;
257 527 753 210 463;
896 896 214 258 259;
532 302 501 765 988;

• Car2（13×4）

654 147 345 447;
321 520 789 702;
12 147 630 255;
345 586 214 866;
678 532 275 332;
963 145 302 225;
25 24 142 589;
874 517 24 996;
114 896 520 541;
785 543 336 234;
203 210 699 784;
696 784 855 512;
302 512 221 345;

• Car3(12×5)

```
456 537 123 214 234;
789 854 225 528 123;
876 632 588 896 456;
543 145 669 325 789;
210 785 966 147 876;
123 214 332 856 543;
456 752 144 321 210;
789 143 755 427 123;
876 698 322 546 456;
543 532 100 321 789;
210 145 114 401 876;
124 247 753 214 543;
```

• Car4(14×4)

```
456 856 963 696;
789 930 21 320;
630 214 475 142;
214 257 320 753;
573 896 124 214;
218 532 752 528;
653 142 147 653;
214 547 532 214;
204 865 145 527;
785 321 763 536;
696 124 214 214;
532 12 257 528;
12 345 854 888;
457 678 123 999;
```

• Car5(10×6)

```
333 991 996 123 145 234;
333 111 663 456 785 532;
252 222 222 789 214 586;
222 204 114 876 752 532;
255 477 123 543 143 142;
555 566 456 210 698 573;
558 899 789 124 532 12;
888 965 876 537 145 14;
889 588 543 854 217 527;
999 889 210 632 451 856;
```

• Car6(8×9)
887 447 234 159 201 555 463 456 753;
799 779 567 267 478 444 123 789 21;
999 999 852 483 520 120 456 630 427;
666 666 140 753 145 142 789 258 520;
663 25 222 120 699 578 876 741 142;
333 558 558 159 875 965 543 36 534;
222 886 965 25 633 112 210 985 157;
114 541 412 863 222 25 123 214 896;

• Car7(7×7)
692 310 832 630 258 147 255;
581 582 14 214 147 753 806;
175 175 785 578 852 2 699;
23 196 696 214 586 356 877;
158 325 530 785 325 565 412;
796 874 214 236 896 898 302;
542 205 578 963 325 800 120;

• Car8(8×8)
456 654 852 145 672 425 214 654;
789 123 369 678 581 396 123 789;
654 123 632 965 175 325 456 654;
321 456 581 421 32 147 789 123;
456 789 472 365 536 852 651 123;
789 654 586 824 325 12 321 456;
654 321 520 758 863 152 456 789;
789 147 120 639 21 863 789 654;

附录 2.2　Hel 类(2 个子问题)

• Hel1(100×10)
1 1 1 4 3 5 5 7 6 4 ;
2 5 4 3 1 9 5 4 7 0 ;
5 6 8 4 4 2 5 6 7 5 ;
4 1 5 6 5 7 9 2 6 2 ;
4 4 2 7 3 6 5 2 4 1 ;
7 6 2 5 4 1 4 7 5 5 ;
8 5 8 7 9 5 3 5 1 5 ;

• Hel1(100×10)
4 2 5 8 9 9 4 7 5 8;
2 7 4 2 5 4 5 8 4 3;
6 5 1 9 4 4 7 6 3 1;
5 4 7 3 9 1 4 7 3 2;
2 4 9 2 4 5 2 1 4 2;
4 0 1 2 2 3 1 4 2 8;
1 2 5 7 8 6 2 1 4 8;
6 4 5 1 2 4 5 6 2 9;
4 5 3 1 8 7 0 1 4 6;
7 3 1 4 7 0 4 1 5 6;
5 2 4 1 2 7 5 3 2 3;
8 6 8 5 7 4 2 5 9 5;
4 5 3 5 7 9 2 4 5 8;
3 5 7 9 6 2 4 4 7 3;
0 2 4 5 4 7 4 5 4 8;
4 2 5 7 4 5 3 2 8 5;
7 8 2 1 9 6 7 8 4 1;
4 8 5 2 6 8 9 5 8 5;
4 5 7 2 3 7 3 6 5 4;
4 2 1 5 1 3 5 6 5 5;
5 8 5 7 8 2 5 8 3 5;
5 4 5 4 5 7 6 2 5 9;
8 2 1 5 5 6 3 8 7 5;
8 3 5 9 5 4 5 2 4 2;
8 5 2 5 7 6 2 8 9 5;
3 7 4 6 8 2 4 5 2 3;
5 5 4 7 9 8 2 5 2 5;
5 2 5 2 5 4 8 2 1 3;
5 5 9 5 4 9 8 5 3 5;
2 1 2 1 4 3 3 5 2 6;
8 8 4 7 2 6 8 6 3 5;
9 7 5 8 5 6 5 8 9 4;
5 6 9 6 5 3 1 8 7 4;
6 4 7 4 3 6 1 4 5 8;
4 3 7 5 1 9 2 4 2 5;
4 2 8 7 3 4 9 8 7 4;
2 5 9 4 2 5 3 0 4 7;
9 5 4 2 3 7 0 2 1 6;
2 3 2 5 1 0 8 9 5 3;
5 2 7 9 4 3 6 2 5 0;
7 8 2 1 4 7 5 8 9 4;
1 4 2 3 6 8 2 4 7 5;
2 5 4 3 6 8 4 1 7 5;

• Hell(100×10)

8 3 0 2 5 6 8 2 9 1 ;
7 2 4 3 6 2 9 4 1 8 ;
3 5 7 5 3 8 6 4 8 1 ;
5 0 5 6 0 0 2 4 7 8 ;
1 9 5 2 4 7 5 0 2 5 ;
0 2 9 6 1 4 0 0 5 2 ;
0 2 5 8 3 6 9 1 2 4 ;
7 9 6 3 5 1 7 5 4 5 ;
1 3 5 2 1 4 9 7 4 1 ;
0 3 5 2 1 9 4 7 5 4 ;
7 8 5 6 3 9 8 7 4 6 ;
1 9 6 7 0 2 4 8 3 6 ;
6 1 2 0 3 5 4 1 7 3 ;
6 5 1 1 9 7 3 5 6 4 ;
1 8 2 6 9 4 7 5 8 4 ;
0 1 6 2 9 4 8 5 7 6 ;
1 2 5 6 8 5 6 4 1 4 ;
3 4 5 8 4 1 2 3 6 8 ;
9 8 2 3 1 4 0 2 4 5 ;
4 3 2 5 6 4 1 8 9 2 ;
5 7 1 2 6 8 2 3 4 7 ;
2 1 4 3 8 4 6 2 4 5 ;
6 7 9 2 4 3 2 5 6 7 ;
2 4 0 2 5 3 1 7 8 6 ;
2 7 5 4 3 1 5 6 0 2 ;
3 5 7 0 2 4 5 2 5 7 ;
2 4 5 3 4 7 8 3 2 4 ;
9 9 1 4 5 7 6 5 3 2 ;
8 2 5 2 2 5 1 5 7 8 ;
4 5 2 4 7 9 5 4 2 4 ;
5 5 1 2 4 2 3 8 5 1 ;
0 2 9 5 4 2 5 9 6 5 ;
1 2 3 5 6 2 4 0 2 5 ;
4 2 3 5 4 2 3 5 4 2 ;
4 4 5 8 9 8 5 2 8 3 ;
4 2 3 3 2 5 8 8 1 2 ;
5 3 6 2 5 6 4 7 9 3 ;
4 2 3 6 8 5 3 4 7 2 ;
5 8 8 3 5 6 5 6 5 2 ;
5 6 4 2 5 4 6 5 8 4 ;
2 1 4 7 4 5 9 8 5 6 ;
2 1 4 6 5 8 6 1 3 5 ;
9 5 1 3 5 7 9 1 2 5 ;

• Hel1(100×10)
3 5 4 9 7 2 6 5 2 1;
2 5 0 3 2 4 7 8 9 5;
5 3 5 7 9 2 4 5 5 8;
6 3 1 5 0 1 4 8 9 8;
3 0 4 3 7 2 6 9 4 1;
1 7 4 2 2 4 5 0 6 9;
1 7 8 4 6 5 4 8 5 2;

• Hel2(20×10)
1 1 1 4 3 5 5 7 6 4;
2 5 4 3 1 9 5 4 7 0;
5 6 8 4 4 2 5 6 7 5;
4 1 5 6 5 7 9 2 6 2;
4 4 2 7 3 6 5 2 4 1;
7 6 2 5 4 1 4 7 5 5;
8 5 8 7 9 5 3 5 1 5;
4 2 5 8 9 9 4 7 5 8;
2 7 4 2 5 4 5 8 4 3;
6 5 1 9 4 4 7 6 5 1;
5 4 7 3 9 1 4 7 3 2;
2 4 9 2 4 5 2 1 4 2;
4 0 1 2 2 3 1 4 2 8;
1 2 5 7 8 6 2 1 4 8;
6 4 5 1 2 4 5 6 2 9;
4 5 3 1 8 7 0 1 4 6;
7 3 1 4 7 0 4 1 5 6;
5 2 4 1 2 7 5 3 2 3;
8 6 8 5 7 4 2 5 9 5;
4 5 3 5 7 9 2 4 5 8;

附录 2.3 Rec 类(21 个子问题)

• Rec01(20×5)	• Rec03(20×5)
5 76 74 99 26;	34 6 63 85 60;
74 21 83 52 90;	11 65 4 1 73;
67 48 6 66 38;	63 67 3 73 100;
97 36 71 68 81;	22 46 88 1 66;
87 86 64 11 31;	76 4 34 9 76;
1 42 20 90 23;	20 22 7 3 28;

• Rec01(20×5)	• Rec03(20×5)
69 32 99 26 57;	44 30 55 68 92;
69 12 54 80 16;	29 89 12 96 71;
11 63 24 16 89;	54 12 21 74 2;
87 52 43 10 26;	62 96 61 79 53;
25 59 88 87 40;	50 13 48 40 37;
50 42 72 77 29;	89 69 57 1 70;
58 76 71 82 94;	50 56 8 67 46;
79 48 20 63 97;	32 24 23 87 62;
35 57 78 99 80;	12 88 64 14 13;
70 76 53 2 19;	59 78 95 59 48;
79 22 77 74 95;	41 20 83 65 20;
34 99 49 3 61;	94 48 26 93 3;
37 24 32 35 4;	28 59 10 81 20;
50 88 46 63 76;	66 33 34 8 5;

• Rec05(20×5)
59 37 67 39 30;
89 41 42 59 43;
18 56 75 95 75;
65 67 50 57 13;
1 79 71 78 88;
49 100 30 76 36;
99 9 34 44 62;
35 46 58 26 73;
8 98 97 20 73;
39 73 20 55 30;
60 18 97 61 22;
71 1 4 88 52;
76 30 51 77 22;
30 98 25 43 5;
77 36 76 16 45;
5 82 64 13 14;
98 84 4 34 26;
79 28 84 69 36;
38 36 47 86 1;
69 19 54 83 97;

• Rec07(20×10)	• Rec09(20×10)
28 18 38 11 97 23 90 52 79 63;	77 95 41 97 47 45 10 41 72 8;
50 30 75 82 38 39 28 84 48 57;	99 28 42 4 7 30 65 45 51 94;
75 50 33 58 56 41 51 29 75 97;	74 25 92 29 4 21 47 36 61 9;
65 42 66 29 36 29 10 84 14 67;	4 21 40 80 66 85 1 33 1 4;
84 68 42 41 86 23 95 30 73 97;	49 95 96 74 96 63 59 84 70 29;
33 72 79 85 81 51 72 19 48 48;	53 59 75 19 13 50 82 60 9 13;
91 66 87 88 97 36 21 59 61 4;	88 47 28 11 86 90 93 38 33 59;
51 23 100 93 48 84 74 7 98 55;	92 99 84 13 73 55 19 93 74 25;
58 61 17 54 25 71 52 47 49 86;	2 49 86 46 58 42 24 79 12 77;
44 27 40 19 34 33 3 89 39 66;	97 18 28 77 92 54 49 24 19 71;
70 94 7 19 31 48 38 48 73 34;	28 93 93 7 25 89 49 11 93 45;
60 38 34 55 63 28 70 35 68 88;	64 22 91 56 46 27 32 70 94 5;
39 33 53 87 2 6 51 42 93 67;	25 96 98 51 21 20 93 64 86 11;
72 35 45 20 84 23 10 34 8 48;	19 41 87 15 31 78 54 74 71 6;
100 71 80 89 47 15 90 33 97 26;	81 1 74 56 8 55 3 92 28 5;
79 23 57 54 70 99 85 5 9 4;	9 29 49 48 72 38 26 3 49 80;
14 23 36 79 4 65 78 51 95 79;	5 74 19 27 71 35 52 76 79 47;
3 32 81 26 19 59 80 90 44 33;	8 66 40 71 17 61 84 49 52 56;
68 33 94 37 33 74 64 50 22 17;	34 7 58 94 22 27 40 19 26 77;
94 17 54 27 55 34 7 56 10 41;	13 56 45 27 40 26 90 28 27 88;

• Rec11(20×10)
25 44 10 41 64 52 19 28 72 27;
76 62 48 54 47 35 72 54 27 56;
41 6 3 33 77 44 43 50 19 43;
65 91 75 30 47 55 51 1 36 73;
81 69 65 93 61 3 44 17 6 14;
19 9 12 54 75 66 34 12 32 6;
93 89 31 14 37 57 33 96 32 45;
39 83 55 32 18 9 93 65 75 73;
52 46 64 13 54 62 45 80 19 65;
72 4 29 94 85 51 29 65 50 16;
55 43 47 32 87 97 41 86 17 30;
8 91 81 93 14 86 64 42 70 3;
27 11 94 38 33 67 8 55 99 18;
34 86 87 10 64 30 47 51 69 26;
15 5 39 23 16 1 57 55 62 35;
59 55 43 49 23 25 51 72 9 1;
93 4 43 5 84 55 22 78 31 11;
20 91 73 41 100 38 75 9 76 71;
59 13 93 26 11 7 66 42 54 99;

• Rec13(20×15)	• Rec15(20×15)
78 37 79 98 100 21 82 27 7 22 9 57 84 91 5;	60 70 51 74 6 4 27 5 30 79 58 49 91 20 63;
13 81 100 77 45 39 60 87 38 91 17 85 43 81 33;	79 53 74 87 18 16 44 95 41 3 20 59 74 33 42;
47 9 31 40 86 27 69 50 87 34 13 15 95 96 72;	94 57 19 77 70 90 36 73 37 24 28 23 16 97 75;
26 61 5 22 26 52 57 97 10 68 8 49 41 16 35;	23 3 82 87 45 54 76 60 61 98 68 14 10 17 9;
68 41 46 58 37 59 22 43 49 21 42 70 13 2 76;	38 32 68 7 49 49 72 31 70 8 26 81 86 60 74;
47 60 23 23 57 60 35 56 54 73 81 61 15 70 51;	27 14 77 20 92 74 33 44 43 52 10 81 30 93 71;
63 21 10 50 12 84 91 7 44 75 60 72 28 83 52;	23 55 67 6 64 68 19 73 92 96 3 21 51 3 25;
52 100 38 79 90 52 81 54 51 11 76 50 24 12 23;	41 91 90 43 37 96 99 46 41 51 33 49 17 64 68;
91 21 59 55 39 75 3 11 14 24 36 72 48 69 55;	92 44 12 37 75 20 14 43 26 64 85 14 54 54 97;
100 54 77 53 90 91 53 68 18 86 28 61 86 36 15;	69 69 71 54 36 96 3 27 16 61 94 96 85 34 34;
63 58 93 24 18 78 74 57 100 92 51 73 6 12 83;	16 38 36 48 92 20 55 89 75 57 89 62 36 36 69;
96 60 79 84 86 31 41 79 63 17 31 65 41 77 74;	37 19 55 2 96 10 81 29 2 34 83 1 45 23 64;
54 55 89 77 40 71 21 43 60 58 13 3 73 44 85;	4 57 26 29 91 94 21 30 30 51 31 62 53 29 51;
50 33 51 100 46 27 30 80 56 50 97 70 9 22 92;	55 65 21 99 27 53 42 78 63 25 86 60 64 76 69;
17 7 12 66 98 25 76 25 76 25 69 69 73 45 74;	17 70 33 81 24 35 77 21 83 19 44 70 90 34 90;
46 76 32 32 28 48 63 92 85 69 24 38 57 21 66;	76 94 54 34 12 28 24 5 59 4 46 93 32 78 48;
91 42 30 39 97 72 96 78 71 75 17 26 94 3 39;	81 49 83 72 31 72 91 8 40 93 84 63 67 36 84;
99 93 57 23 52 89 4 74 21 10 53 94 59 100 68:	63 70 14 85 66 41 64 54 10 65 26 58 20 64 41;
20 84 25 1 72 63 79 63 17 38 21 36 1 73 58;	98 39 91 5 55 28 43 97 35 23 50 99 59 63 95;
18 4 21 43 55 86 99 38 81 74 34 40 61 76 100:	47 48 89 33 99 10 54 94 52 44 73 67 44 36 98;

• Rec17(20×15)
59 11 4 79 94 31 74 82 53 51 19 31 46 47 10;
72 54 36 3 59 23 40 59 89 37 85 67 39 65 60;
92 26 15 81 86 56 92 47 93 21 40 77 84 10 91;
49 27 99 64 30 51 26 89 40 64 60 67 67 100 3;
42 12 55 62 37 24 24 42 41 88 14 33 85 4 20;
21 61 52 49 44 98 26 68 61 25 6 46 75 37 5;
80 99 88 83 11 93 47 80 100 87 84 17 43 93 58;
4 54 43 63 44 78 44 39 76 99 29 38 14 75 25;
46 23 54 77 60 53 42 72 90 11 22 68 94 24 14;
23 84 92 94 8 10 77 58 64 95 55 15 19 62 67;
53 91 80 8 41 89 3 87 57 75 37 8 23 88 65;
72 17 53 36 9 24 80 9 28 60 94 99 67 10 44;
5 44 96 37 21 44 49 13 86 74 89 3 82 85 61;
80 19 73 95 78 78 31 13 50 93 98 80 46 9 37;
7 16 46 82 97 82 41 21 11 50 5 28 95 84 45;
68 73 57 4 66 71 87 43 60 56 30 21 14 37 61;
77 2 1 82 2 49 89 27 34 52 85 26 80 87 58;
17 86 32 35 6 50 53 39 94 89 22 75 59 74 27;
61 14 70 58 24 36 70 57 31 100 21 76 54 94 57;
29 6 6 12 78 28 40 13 61 19 39 98 69 14 3;

• Rec19(30×10)	• Rec21(30×10)
40 16 50 59 100 78 38 76 9 68;	12 89 69 94 22 31 22 95 36 38;
39 3 35 70 65 80 40 49 52 50;	55 2 94 19 43 74 72 10 99 16;
31 56 88 71 83 69 48 98 88 96;	34 78 27 40 21 17 15 62 96 63;
65 77 58 66 86 93 69 49 85 51;	100 14 23 42 52 18 29 70 21 47;
11 5 75 12 56 64 20 6 83 49;	41 92 88 52 99 41 68 22 57 66;
48 56 41 55 3 94 11 87 78 48;	24 62 19 35 24 49 100 65 13 41;
47 60 16 67 61 36 36 62 13 74;	29 34 38 72 29 83 44 91 65 100;
67 100 3 85 70 19 58 87 61 51;	89 62 32 54 93 59 8 24 86 66;
32 47 40 47 66 85 99 50 19 45;	23 34 89 66 10 48 27 11 94 45;
27 97 84 30 68 28 26 98 88 96;	81 57 35 78 23 66 1 3 77 14;
81 12 1 88 63 32 38 82 68 61;	72 47 75 27 66 64 30 49 42 7;
58 56 53 88 100 8 57 92 39 45;	10 15 26 12 98 12 53 81 46 3;
38 30 81 51 70 28 10 93 53 45;	47 54 58 73 44 87 87 98 34 15;
14 13 1 84 97 69 20 68 19 83;	13 27 29 85 29 64 62 62 79 74;
70 98 83 22 27 44 93 46 91 45;	49 57 24 44 18 97 59 75 17 22;
75 30 45 64 13 47 6 49 57 21;	50 11 93 53 52 13 51 76 87 95;
69 41 37 12 3 81 92 25 24 36;	99 87 85 9 87 98 34 22 66 11;
47 92 28 4 28 3 32 85 8 94;	7 8 90 95 29 79 70 79 6 58;
52 7 97 56 90 60 37 42 19 15;	48 100 74 60 74 19 21 6 77 84;
58 11 18 100 47 24 41 48 51 65;	96 86 19 15 45 1 90 49 98 80;
61 69 45 17 4 31 83 32 68 5;	68 73 55 13 28 16 57 20 76 71;
63 22 5 77 99 19 99 37 92 19;	53 38 4 43 11 49 12 91 47 3;
63 25 83 78 89 66 8 57 89 56;	57 13 12 12 21 68 2 80 9 28;
42 86 8 83 39 26 99 75 60 67;	37 79 92 35 63 13 58 36 65 94;
51 100 42 53 10 66 19 2 24 41;	39 49 57 23 53 80 42 29 52 33;
100 90 68 91 46 5 59 11 10 44;	36 54 59 69 62 12 77 37 87 47;
42 41 76 76 61 52 44 78 40 57;	63 35 26 38 47 82 89 34 1 93;
83 24 14 100 26 41 19 18 21 12;	60 28 1 51 94 86 42 75 76 77;
5 84 57 6 60 91 18 83 44 87;	2 51 79 74 51 28 78 87 81 35;
69 35 72 62 90 8 44 67 4 77;	45 94 42 9 70 4 52 54 16 27;

• Rec23(30×10)
54 2 2 75 39 97 70 5 40 16;
51 69 31 100 25 58 30 3 18 64;
44 18 6 6 9 96 52 31 52 14;
66 95 42 1 21 65 24 78 20 81;
18 53 28 61 84 52 33 31 14 35;
100 39 72 19 44 66 8 59 80 94;
51 86 28 56 13 22 51 99 51 8;
88 95 45 83 5 30 81 19 86 46;
41 24 93 33 70 84 66 50 11 87;
92 78 9 47 39 48 65 79 19 4;

• Rec23(30×10)
12 95 32 60 53 44 63 28 66 53;
98 31 39 91 7 38 10 2 4 60;
11 53 83 29 85 31 55 100 16 10;
16 22 32 21 96 42 88 100 12 31;
13 80 5 57 15 99 76 36 85 87;
83 41 11 64 16 26 98 77 59 60;
17 78 85 87 16 29 88 16 20 41;
59 8 26 77 4 84 64 62 56 14;
44 79 34 85 86 25 76 21 94 49;
31 63 50 39 20 32 34 9 39 85;
44 37 88 43 50 17 52 38 8 17;
9 74 81 55 50 13 82 100 69 89;
90 31 8 79 55 59 52 59 83 75;
29 33 42 54 5 93 5 38 32 70;
81 66 79 36 75 32 36 2 68 77;
53 58 82 91 21 65 28 53 39 95;
40 72 13 79 9 39 90 9 37 3;
66 90 75 45 59 46 98 99 26 55;
16 3 46 2 79 11 26 47 88 58;
78 97 19 22 5 79 90 57 54 68;

• Rec25(30×15)	• Rec27(30×15)
68 91 17 68 46 86 24 43 58 86 40 27 38 82 86;	94 24 34 45 45 93 28 83 12 73 34 64 57 58 49;
34 21 4 48 67 24 63 68 96 45 91 97 96 3 46;	73 46 90 94 20 9 2 89 35 38 87 14 15 28 50;
57 69 12 2 51 68 34 8 17 55 80 61 51 32 36;	55 6 28 38 93 74 81 89 61 38 9 61 90 24 65;
22 24 24 3 76 65 94 69 73 33 86 36 48 42 85;	53 9 67 19 46 41 68 33 68 46 51 96 74 16 71;
28 6 47 71 81 93 94 21 32 23 73 48 35 67 59;	89 63 14 77 14 47 3 35 98 96 24 45 68 18 23;
79 80 27 21 56 36 24 94 53 50 55 7 78 14 53;	41 42 58 67 92 21 100 39 85 42 37 10 41 16 67;
98 55 71 80 23 45 44 22 40 93 38 4 96 42 53;	80 35 48 72 77 19 75 37 81 48 64 2 1 65 4;
69 84 63 15 27 66 73 98 64 38 3 69 46 27 34;	63 83 93 11 87 67 38 33 43 40 46 3 86 1 68;
65 21 79 50 67 68 56 53 71 29 63 36 62 77 35;	84 78 16 19 31 56 56 18 77 65 82 14 96 20 6;
9 92 84 88 48 71 71 90 24 54 77 96 66 49 29;	10 94 30 17 37 50 14 50 34 80 88 52 100 37 65;
92 22 73 4 38 31 55 49 66 83 75 82 87 82 89;	36 32 59 18 56 39 65 77 91 21 58 90 65 33 65;
27 67 89 10 35 67 88 43 51 22 23 60 54 22 76;	66 98 49 11 61 85 65 8 46 56 39 29 63 42 4;
10 65 77 85 5 25 14 78 32 23 21 11 65 60 23;	79 42 65 49 77 12 38 75 80 76 65 50 8 9 33;
56 74 66 61 27 41 100 26 92 79 100 39 11 59 97;	90 52 18 17 66 7 83 24 37 48 15 92 72 20 2;
93 96 79 50 35 28 100 84 78 81 65 69 17 96 19;	64 40 19 49 100 74 12 71 80 9 31 1 55 18 9;
2 99 100 13 15 35 3 58 39 56 57 48 82 86 53;	34 69 36 47 71 44 71 20 5 96 15 83 93 36 95;
2 72 49 44 84 48 90 48 27 45 49 26 36 20 33;	25 6 13 9 46 80 37 52 76 95 56 84 17 9 51;
86 54 79 52 2 67 69 78 38 92 13 25 40 37 80;	40 62 98 3 23 69 1 3 43 68 91 37 92 84 48;
51 99 68 2 66 44 98 83 50 53 13 57 39 50 92;	6 38 74 83 21 14 84 41 100 88 58 59 36 99 68;

• Rec25(30×15)	• Rec27(30×15)
38 25 18 97 35 7 45 98 81 18 60 73 86 34 3；	17 79 95 70 70 40 53 5 86 57 59 37 40 33 89；
4 29 84 11 61 47 17 2 68 85 93 64 98 34 62；	55 15 11 12 85 40 11 26 8 97 38 28 3 59 30；
23 95 66 57 91 15 90 84 25 88 65 24 80 98 76；	84 59 92 29 89 89 56 48 32 41 29 9 58 84 58；
81 43 11 95 78 1 87 11 26 80 29 100 28 40 37；	44 76 34 98 87 99 19 89 43 8 55 55 87 22 97；
54 33 39 23 44 32 16 96 29 87 34 25 80 14 83；	92 97 2 59 47 69 47 85 66 69 19 85 32 86 65；
56 58 10 92 95 95 73 83 57 83 24 54 48 81 20；	71 90 69 8 47 89 53 86 97 93 56 58 65 37 94；
81 24 37 21 97 60 25 21 53 34 57 12 34 28 87；	20 94 66 75 18 14 32 66 2 1 10 9 8 28 66；
55 33 74 43 66 65 32 96 29 7 33 78 30 36 45；	48 56 18 96 53 5 29 22 41 83 57 14 61 90 4；
2 67 56 82 49 74 97 11 75 76 65 41 76 9 80；	48 46 80 26 42 68 81 11 1 41 34 59 33 43 78；
6 88 52 32 54 20 10 7 64 14 35 81 92 22 81；	91 5 10 66 64 57 13 38 26 52 89 38 93 93 50；
88 65 62 51 52 89 23 55 63 79 63 94 79 39 50；	57 78 60 30 20 3 68 83 45 42 94 26 32 47 94；

• Rec29(30×15)
19 84 11 52 100 59 66 18 72 65 96 71 9 23 61；
84 3 50 48 50 5 53 35 57 39 29 17 23 61 99；
85 77 46 60 22 21 53 61 43 1 96 16 25 100 4；
14 44 9 71 29 10 9 79 82 33 63 64 85 39 67；
57 76 4 70 38 58 83 39 36 75 85 40 50 34 35；
43 47 21 8 36 90 84 82 99 69 51 84 100 27 59；
24 47 29 62 41 28 35 60 4 23 29 82 63 53 31；
9 55 99 52 83 65 31 67 85 75 51 36 12 19 44；
46 84 94 86 72 23 20 63 81 17 60 56 10 15 85；
99 35 34 26 44 15 98 99 23 74 93 66 85 56 24；
77 41 4 25 26 15 69 17 86 39 34 97 43 43 28；
11 52 9 71 36 78 82 95 88 88 3 58 63 66 19；
75 31 5 90 80 61 30 14 30 81 83 95 93 7 17；
41 27 97 6 82 80 6 59 40 38 73 22 98 44 23；
4 1 2 3 33 6 35 18 79 78 62 23 24 15 5；
22 51 78 74 96 1 58 51 17 40 37 58 80 64 61；
93 54 86 78 59 10 59 31 37 67 92 74 28 28 40；
46 31 7 95 99 48 87 17 9 26 20 20 96 20 27；
5 47 33 99 83 54 79 7 63 5 67 10 1 27 1；
94 5 39 96 60 56 26 19 49 48 50 31 68 11 83；
97 15 19 30 19 75 32 85 24 79 1 53 81 57 8；
83 28 40 81 25 99 3 91 99 94 7 51 44 10 100；
24 15 53 49 21 32 46 95 57 96 65 61 82 62 92；
78 9 17 74 90 21 85 69 2 49 40 98 90 51 20；
58 28 3 30 40 64 36 23 4 7 17 37 81 96 20；
2 65 45 41 1 96 36 39 55 14 17 20 3 21 67；
82 92 22 44 92 51 4 35 67 80 32 91 49 17 20；
57 60 51 44 30 87 24 42 29 34 79 68 15 78 98；
17 7 9 54 31 17 2 66 47 73 1 16 32 86 85；
79 93 83 8 81 61 69 99 34 68 51 46 91 52 50；

• Rec31(50×10)	• Rec33(50×10)
59 47 20 43 49 74 38 46 18 12;	58 62 99 63 18 68 14 45 98 37;
30 1 90 97 5 70 59 63 15 93;	38 49 31 36 13 39 64 58 72 31;
22 58 68 3 33 48 27 12 65 21;	16 53 33 21 85 99 20 85 96 99;
70 81 2 32 72 57 32 25 13 87;	32 59 13 99 61 98 30 14 22 66;
38 17 48 53 57 17 25 50 72 72;	10 74 78 58 39 23 10 78 75 42;
51 15 72 8 34 90 40 44 47 77;	96 70 92 53 1 52 58 71 15 50;
63 84 75 75 71 13 10 97 81 31;	8 72 78 8 7 94 66 32 90 92;
48 62 71 70 6 94 10 71 29 99;	25 19 53 65 83 57 95 71 68 65;
92 29 91 99 54 64 89 89 38 87;	42 93 72 85 35 68 93 4 94 45;
91 21 56 49 43 20 27 68 99 73;	37 62 60 66 39 71 5 27 65 87;
62 6 3 89 48 97 79 21 96 77;	92 78 25 97 4 79 19 67 90 53;
67 83 70 19 50 50 60 28 15 50;	78 73 64 47 68 47 79 97 37 10;
73 18 55 49 66 56 90 29 87 4;	35 34 89 92 97 74 91 21 41 18;
27 94 71 33 31 68 45 52 95 40;	47 3 73 85 27 28 45 34 29 85;
48 28 46 73 89 35 98 97 67 9;	47 63 36 23 8 3 57 68 43 68;
7 51 48 4 29 62 37 15 10 66;	2 100 92 8 67 27 15 34 63 20;
55 46 65 48 61 36 69 14 78 100;	93 15 96 40 84 59 29 40 15 10;
4 4 31 49 28 78 73 26 29 26;	38 62 29 19 32 13 21 96 70 76;
19 97 37 30 37 16 15 89 11 16;	70 1 63 47 20 79 38 3 60 48;
30 95 86 22 17 16 61 79 24 9;	14 38 71 55 100 41 93 11 20 3;
71 39 93 87 38 7 24 1 91 34;	60 26 23 50 32 93 79 85 48 20;
83 40 37 25 68 47 81 62 96 19;	32 13 38 22 37 59 85 80 94 19;
45 33 12 63 32 40 60 54 66 92;	25 52 62 40 76 15 36 7 18 28;
40 67 83 11 62 69 46 93 80 50;	77 2 38 83 33 60 64 2 33 9;
100 6 82 78 5 43 18 73 86 62;	68 65 85 60 97 25 57 71 94 59;
97 75 81 22 38 2 53 44 73 74;	92 57 17 25 68 26 86 49 74 35;
74 89 14 33 11 43 70 58 47 8;	46 66 7 11 13 66 51 26 41 94;
73 46 62 27 63 34 58 91 11 80;	74 85 55 42 98 51 40 69 20 64;
14 24 27 62 72 85 98 99 25 7;	83 56 31 98 24 75 59 53 50 45;
100 29 8 55 88 96 23 98 19 79;	72 90 55 34 90 21 76 60 82 76;
53 11 74 66 94 66 98 87 5 85;	13 8 2 30 94 24 77 71 100 80;
68 77 88 47 51 73 16 17 87 96;	80 56 40 18 87 98 100 41 2 86;
69 40 46 62 23 31 45 21 15 40;	54 59 82 81 33 71 4 87 21 18;
34 41 10 17 25 33 17 28 45 68;	7 20 10 45 96 70 93 77 48 22;
7 26 79 76 35 92 77 15 27 69;	100 61 16 67 63 96 67 12 20 64;
47 31 83 28 92 83 96 18 84 45;	5 42 57 26 24 35 53 67 94 72;
54 96 8 28 94 50 20 28 99 65;	23 74 65 36 88 68 100 69 84 93;
9 13 81 1 94 82 29 82 27 45;	22 74 74 80 40 65 53 86 19 70;
64 22 51 33 9 25 22 64 78 88;	17 8 16 97 67 95 70 45 17 97;
38 25 16 24 62 4 39 77 36 60;	78 1 55 3 68 26 33 5 9 66;
72 6 40 56 23 39 38 5 75 44;	12 58 9 83 57 27 78 14 61 55;
26 33 37 84 61 86 22 94 93 17;	71 3 68 81 61 3 14 75 49 94;
88 39 63 43 98 27 32 20 25 25;	37 62 60 85 31 84 80 75 59 6;
73 70 57 5 100 31 34 11 98 76;	54 75 14 19 32 66 3 69 77 90;
77 4 85 50 9 45 35 3 41 80;	35 58 3 47 23 98 18 26 64 22;
20 36 9 89 4 32 76 20 84 6;	91 14 89 62 52 59 44 62 76 33;
99 64 7 68 67 85 60 23 55 52;	40 47 1 6 19 1 86 37 79 95;
13 7 80 57 22 78 75 17 70 55;	1 23 77 76 71 99 40 14 27 46;
40 87 34 96 27 78 53 40 72 91;	57 69 76 55 56 83 12 26 12 53;
77 8 14 76 19 82 86 21 10 51;	75 44 90 73 4 74 51 25 20 95;

• Rec35(50×10)

100 72 76 100 16 9 5 87 34 15;
19 3 19 68 29 22 16 13 87 70;
70 56 39 71 29 91 100 86 88 99;
50 93 100 71 84 64 67 29 28 81;
80 97 3 10 14 32 92 67 72 68;
47 59 29 3 26 20 50 26 1 70;
40 63 69 21 56 73 56 10 46 40;
84 80 68 82 4 45 100 96 29 67;
85 46 59 35 68 84 89 18 97 58;
60 60 2 50 90 20 78 56 62 27;
78 64 21 5 85 55 15 23 36 87;
98 31 42 73 83 48 71 49 72 30;
4 57 30 11 67 4 82 77 98 21;
45 45 25 45 7 59 88 12 57 81;
73 94 83 59 1 72 65 62 45 76;
77 84 11 82 10 9 67 27 43 8;
22 66 5 77 97 28 61 82 62 96;
90 51 87 27 65 76 67 20 75 67;
12 92 43 21 92 64 94 67 60 46;
9 76 62 46 71 65 76 65 30 38;
29 12 71 70 46 96 12 70 76 19;
83 15 73 32 51 6 3 29 3 24;
83 95 87 29 46 67 89 73 69 33;
83 46 82 2 55 54 85 3 20 57;
11 32 15 27 2 43 23 79 28 29;
10 74 73 99 54 89 83 5 28 90;
73 40 4 20 51 18 37 18 61 75;
85 30 58 89 48 15 82 77 2 3;
56 63 26 87 53 8 80 46 5 62;
59 67 73 65 60 61 94 86 38 1;
70 66 80 32 93 56 26 41 21 9;
4 66 79 43 39 83 55 25 62 13;
51 42 90 85 84 29 73 8 95 57;
18 30 61 67 57 60 25 10 20 95;
61 9 3 2 61 18 44 78 38 74;
25 91 31 2 14 97 91 84 88 26;
84 8 95 61 85 41 88 4 86 51;
74 2 21 42 33 24 62 13 62 10;
33 7 62 68 42 41 78 67 99 6;
38 43 2 4 62 95 76 91 67 78;
43 98 28 51 43 84 13 71 64 81;
15 19 50 30 75 90 94 35 51 83;
75 98 42 67 24 63 15 45 92 44;
29 60 80 86 70 13 100 86 88 6;
14 49 78 93 45 94 35 46 18 85;
29 20 27 66 70 95 7 11 75 52;
73 19 33 36 93 21 44 51 4 24;
87 79 52 85 24 89 50 4 37 50;
86 99 31 25 78 10 41 66 35 1;
2 41 41 88 6 77 89 80 21 54;

• Rec37(75×20)

3 64 92 79 4 67 69 89 50 9 2 7 42 87 4 93 7 21 14 29;
71 63 89 2 95 21 27 56 31 17 24 5 56 22 71 83 71 31 80 47;
4 39 31 64 41 85 30 58 81 92 19 59 78 7 62 74 7 99 47 35;
33 3 86 65 46 44 23 69 81 95 5 16 63 77 70 88 46 35 82 93;
13 90 44 27 42 10 72 88 34 13 23 51 91 95 12 76 41 77 78 45;
23 62 25 100 81 17 68 36 70 55 41 62 67 46 30 10 72 84 52 37;
68 4 69 56 45 29 30 64 44 59 50 72 19 40 92 88 45 43 46 45;
77 27 44 95 37 29 100 45 36 85 69 30 3 15 90 35 88 64 27 66;
6 68 42 39 87 29 44 76 51 18 39 9 99 30 40 98 57 84 87 24;
34 87 81 37 51 66 56 81 17 70 27 8 10 71 86 51 31 64 44 67;
65 70 2 63 7 99 51 9 90 77 26 50 68 50 93 92 15 88 40 68;
29 28 73 14 60 15 72 34 50 100 53 83 26 3 55 45 38 70 42 27;
82 97 86 15 41 25 87 53 1 25 50 81 64 90 89 58 56 43 42 67;
2 38 58 14 3 65 88 90 72 62 84 43 87 98 36 5 48 70 57 35;
94 37 31 83 26 96 92 94 19 47 48 91 24 37 64 85 18 36 30 6;
89 5 72 23 16 3 40 37 21 42 58 75 31 43 66 19 96 52 27 78;
40 78 75 55 2 98 1 19 83 33 57 67 87 78 58 58 38 54 41 74;
28 16 55 29 38 16 66 63 35 30 12 67 96 27 82 74 22 3 90 43;
57 6 55 18 4 19 53 17 31 69 15 97 88 100 6 66 54 84 14 58;
56 59 42 20 5 92 3 54 51 11 83 6 19 35 22 58 62 10 83 17;
82 10 9 59 71 47 84 74 99 42 59 68 67 92 47 97 3 16 86 45;
12 21 72 83 56 19 89 100 6 74 10 80 38 57 23 57 54 62 43 89;
52 77 74 66 40 69 34 79 25 42 27 67 51 40 18 37 22 35 95 24;
32 90 82 7 27 83 61 86 38 6 29 76 90 78 23 60 36 45 8 46;
6 94 88 27 48 73 38 19 47 54 56 82 6 33 56 75 50 29 77 81;
53 52 10 77 56 73 57 1 84 95 78 82 13 88 59 54 45 48 16 75;
71 46 90 47 12 7 51 38 19 47 46 100 41 57 100 97 24 87 70 3;
50 5 30 86 22 100 10 43 57 80 11 13 26 35 66 98 23 77 31 26;
30 33 34 5 65 65 30 42 86 12 57 59 40 37 23 33 96 30 62 67;
24 98 2 10 29 43 83 14 69 58 32 92 65 81 30 95 62 76 87 99;
66 93 57 28 24 3 69 66 92 62 9 94 63 84 35 38 58 8 30 52;
17 34 47 16 87 36 10 26 64 91 87 67 18 88 11 43 45 43 40 80;
64 64 45 53 78 59 55 48 22 36 89 43 42 61 57 14 9 22 90 63;
17 51 51 44 82 14 77 14 27 16 64 76 71 95 95 81 60 14 78 6;
12 85 87 18 12 69 75 27 65 72 86 45 66 6 75 98 35 11 28 39;
11 84 3 30 4 17 73 17 47 71 51 55 12 37 71 46 94 29 88 16;
98 91 70 59 17 73 18 38 9 6 85 36 21 56 79 28 23 10 32 17;
86 73 89 11 37 80 85 71 12 94 36 42 29 54 24 75 34 23 58 47;
10 22 82 15 77 6 71 20 41 77 76 54 96 48 55 41 62 59 90 35;
31 55 98 26 72 43 1 15 7 22 5 75 8 81 44 6 69 35 76 36;
35 14 10 23 82 7 21 11 48 22 46 36 77 21 91 75 39 94 89 29;
32 66 16 11 94 42 7 86 22 93 47 91 38 58 18 77 85 2 28 20;
60 4 41 20 19 13 56 47 18 77 81 20 52 11 73 67 42 52 16 38;
78 45 32 4 92 14 92 22 44 43 48 28 5 21 32 25 70 62 2 38;
73 31 63 44 71 62 63 66 38 82 70 44 9 56 91 55 57 72 59 99;
55 30 32 44 52 98 43 59 60 11 96 23 98 8 75 87 12 51 86 26;
61 32 64 33 37 91 30 6 96 73 100 45 35 94 31 75 72 91 6 49;
51 56 5 66 63 31 48 36 57 35 80 14 4 56 34 1 98 9 67 23;
12 44 30 23 93 37 44 4 80 88 59 60 17 61 81 67 76 46 44 27;
54 63 38 9 12 50 93 90 24 51 85 38 10 3 96 4 16 71 33 65;

• Rec37(75×20)

13 5 50 53 31 38 93 98 76 64 33 27 44 97 28 67 55 75 60 68;
51 88 17 10 86 93 67 58 47 96 1 32 4 86 49 34 53 76 72 91;
53 53 12 60 38 32 64 25 73 39 11 84 2 37 59 12 97 15 42 12;
30 67 32 7 73 98 58 79 15 10 13 35 74 43 81 29 97 61 70 37;
18 24 1 82 2 57 45 23 58 51 80 67 80 20 63 41 89 5 78 95;
78 39 69 38 61 1 51 88 14 29 10 84 6 11 79 26 34 63 15 42;
95 56 7 55 10 48 24 29 18 36 28 15 64 83 27 90 19 80 55 55;
82 42 15 39 36 40 36 25 32 90 80 16 89 29 31 62 54 21 95 42;
74 88 16 18 50 45 87 81 41 32 41 65 42 64 38 98 74 79 60 30;
31 11 59 48 87 54 99 26 57 31 9 68 10 32 23 76 42 71 98 75;
40 55 17 8 9 84 22 16 14 78 8 54 2 100 84 1 43 40 82 56;
11 86 88 2 1 78 28 29 75 90 86 79 56 60 44 3 87 52 10 79;
81 97 95 62 74 4 17 71 26 9 16 21 31 28 52 37 11 99 7 72;
10 4 89 40 65 52 11 72 76 4 98 85 15 48 81 83 76 100 19 91;
85 49 40 18 34 41 59 85 62 65 53 2 17 87 30 45 68 95 22 15;
16 1 49 99 67 12 35 91 46 10 31 25 74 36 22 52 97 53 14 49;
40 51 39 14 75 33 36 94 12 82 6 34 47 13 5 60 100 64 17 23;
17 16 57 16 94 95 87 25 72 47 75 35 21 60 53 48 37 48 80 90;
33 39 78 86 85 17 35 37 62 97 62 57 8 44 65 21 95 45 77 32;
98 86 100 29 34 40 55 11 43 51 15 54 44 57 63 9 84 24 6 52;
50 51 46 73 52 32 45 51 100 49 9 6 89 17 97 65 40 75 91 81;
49 49 65 75 75 84 27 86 45 60 41 18 30 66 39 71 25 17 80 72;
81 24 30 12 63 14 51 72 13 75 35 19 76 71 89 30 23 91 46 6;
57 42 39 91 3 64 90 88 66 12 39 65 66 24 69 73 47 65 88 88;
13 96 17 41 6 1 42 40 89 12 26 35 58 43 28 53 32 31 30 47;

• Rec39(75×20)

55 40 64 57 84 85 40 96 68 37 70 48 85 67 43 47 35 69 72 51;
24 12 19 81 46 1 88 66 16 20 41 75 85 16 41 18 31 57 77 26;
81 90 30 12 61 31 69 13 6 51 46 75 43 70 99 81 23 33 94 39;
84 10 89 73 60 86 49 36 44 14 26 21 37 89 36 86 66 33 15 14;
10 34 14 13 97 2 21 59 59 21 51 12 72 61 92 7 38 17 58 99;
64 47 46 6 51 40 2 29 5 7 11 71 78 95 35 40 45 72 65 68;
77 60 53 50 28 86 35 21 63 35 24 47 9 27 58 10 17 8 3 19;
3 8 27 68 57 51 86 32 38 89 32 12 46 19 78 66 8 61 64 76;
22 54 84 9 93 70 48 86 57 81 2 66 92 56 36 80 99 92 65 86;
37 83 68 45 15 6 37 32 78 46 26 32 78 50 100 51 92 88 48 25;
80 6 46 77 28 43 96 43 79 46 22 58 58 36 63 3 19 58 82 88;
5 58 9 27 62 92 93 17 64 66 32 17 12 75 40 84 34 40 57 2;
91 1 75 43 72 90 19 78 64 18 23 9 45 54 76 68 17 94 91 13;
44 21 86 99 36 98 92 74 33 36 87 56 86 82 36 52 72 5 27 95;
92 28 9 93 61 1 53 79 51 83 41 7 90 97 12 61 29 86 31 47;
90 59 37 49 75 10 70 91 23 45 28 36 37 58 23 42 43 71 96 92;
19 75 86 16 33 62 36 34 78 31 14 41 31 52 10 70 93 22 38 78;
83 60 99 66 15 16 67 55 71 78 93 45 2 91 41 42 84 24 1 80;
11 28 43 98 26 60 5 29 82 47 80 96 5 9 7 82 46 90 31 94;
58 12 10 36 95 5 18 54 15 25 19 69 19 67 26 39 34 53 80 46;
4 75 17 28 78 85 98 53 100 21 76 62 50 53 41 85 27 76 6 83;
54 1 7 48 52 62 62 75 92 71 44 97 26 76 17 20 29 45 70 78;
38 3 47 93 25 22 52 94 71 38 40 24 3 72 48 67 72 70 41 31;
9 16 29 87 24 76 36 6 40 6 5 16 59 3 50 74 8 87 9 64;

• Rec39(75×20)

48 1 69 80 5 59 6 37 30 87 8 71 1 55 66 15 18 58 93 26;
59 44 11 43 47 34 78 35 96 16 41 14 57 37 63 88 7 67 35 91;
72 56 21 76 56 85 95 72 17 55 19 91 83 88 33 17 5 27 1 56;
42 72 92 2 60 23 25 48 98 89 87 77 57 66 94 49 65 73 84 46;
49 54 41 70 15 85 58 85 68 30 11 70 85 58 87 59 68 65 1 9;
96 88 10 53 100 32 14 33 83 13 82 94 97 76 80 45 18 91 94 18;
15 84 66 49 20 49 33 64 21 99 20 97 46 54 64 30 45 61 31 71;
58 79 1 81 4 47 83 78 88 74 64 52 13 55 58 63 5 10 4 92;
8 34 33 99 8 63 57 97 14 49 83 58 2 63 3 17 75 100 29 30;
66 34 4 76 10 57 70 70 52 61 70 39 44 91 66 91 62 16 50 58;
65 90 81 10 16 15 66 71 83 69 40 42 92 73 40 8 94 71 32 75;
58 38 56 25 56 48 12 97 12 60 36 42 27 70 42 16 27 98 69 3;
24 40 46 68 83 92 100 73 67 46 26 36 52 70 14 89 40 61 78 91;
69 81 94 14 78 9 47 46 4 60 2 48 99 82 24 26 38 43 100 13;
23 72 66 61 46 83 95 91 3 66 81 26 21 43 62 49 51 57 3 67;
39 49 55 32 16 26 12 5 68 48 4 45 98 13 47 8 32 39 80 77;
99 73 79 76 42 73 89 13 28 18 40 2 35 78 21 75 63 48 47 91;
5 30 78 66 5 87 44 62 39 12 80 21 62 50 51 50 47 72 46 82;
89 30 22 100 8 52 19 26 81 89 42 50 32 64 29 55 15 21 45 49;
4 11 100 3 83 79 81 3 57 38 67 62 27 81 72 39 21 30 36 16;
31 32 32 22 83 6 23 17 97 67 23 57 50 88 23 75 44 62 36 30;
74 80 59 31 88 91 61 16 57 14 3 57 32 94 49 63 90 2 87 65;
63 66 47 29 4 22 38 74 16 39 23 12 27 37 41 25 87 60 57 20;
52 25 91 38 58 10 21 89 79 29 25 94 14 77 18 9 92 74 37 18;
21 18 5 8 7 90 2 85 74 93 77 1 99 99 20 90 82 3 45 8;
75 32 34 11 30 23 99 10 58 69 71 58 11 15 17 65 64 34 23 62;
44 77 44 47 32 95 53 37 9 16 25 13 4 62 99 58 67 77 39 80;
82 90 26 38 41 17 31 65 33 21 79 39 59 98 83 18 45 69 84 85;
69 31 99 33 14 25 27 17 59 95 2 35 80 12 13 18 79 70 75 25;
10 86 5 5 30 80 57 42 41 74 85 23 97 13 55 56 72 67 55 73;
33 66 10 54 92 68 62 14 60 19 47 54 64 38 1 56 55 70 91 29;
95 7 7 1 32 100 12 30 20 15 28 99 62 47 69 66 82 15 75 15;
75 70 13 97 4 12 97 14 51 74 96 57 94 26 1 60 33 64 25 81;
77 41 71 13 87 65 35 14 8 32 44 52 91 87 64 36 13 1 82 98;
31 30 47 49 86 44 68 4 69 49 89 33 7 65 50 18 50 39 14 67;
91 75 33 29 31 61 63 83 41 11 72 72 22 22 76 54 1 13 13 10;
38 35 48 99 76 51 12 74 52 30 61 69 41 42 84 17 45 84 95 76;
77 97 32 35 2 76 33 25 57 41 49 46 93 38 43 6 86 38 4 38;
36 72 54 34 12 21 68 9 36 5 51 52 33 45 43 44 54 85 7 94;
70 95 5 20 37 97 84 26 94 8 12 60 40 23 2 80 4 63 95 69;
60 27 3 41 32 41 73 98 60 62 97 69 20 60 63 87 15 31 86 10;
9 55 89 71 75 13 52 75 89 1 98 3 2 65 92 64 92 76 22 91;
48 90 31 8 72 99 64 28 60 88 34 9 41 76 82 34 65 8 70 10;
32 66 22 76 52 10 75 58 79 7 46 62 16 51 49 45 89 64 23 92;
39 46 78 22 70 83 79 87 76 70 2 59 33 79 37 72 42 4 97 83;
75 15 41 21 6 77 91 63 3 13 93 24 20 69 12 11 31 15 35 58;
69 84 79 71 64 79 55 61 43 97 37 6 32 58 66 88 84 7 4 16;
76 89 84 96 74 100 37 79 98 7 77 77 50 6 17 50 56 17 96 79;
76 91 73 43 91 76 30 39 99 55 78 38 78 100 6 70 28 4 28 95;
73 76 88 86 94 68 51 91 1 77 34 9 45 50 2 48 3 56 43 40;
97 55 97 24 89 61 41 8 83 33 61 42 6 92 95 6 11 83 8 10;

• Rec41(75×20)

88 49 15 75 21 6 79 74 45 63 20 18 13 64 3 13 1 80 15 93;
50 16 58 11 21 82 23 33 2 95 19 36 18 76 87 1 21 71 89 36;
49 45 92 46 99 1 9 37 99 79 38 68 53 73 92 98 86 3 11 10;
66 95 36 29 8 100 32 8 68 25 44 14 23 40 61 80 74 40 98 63;
6 54 31 33 28 40 7 96 62 70 29 97 57 88 61 43 91 65 92 68;
97 31 45 56 65 7 78 74 61 74 11 65 11 54 85 13 67 90 62 24;
95 63 69 23 47 11 10 40 70 24 32 92 63 98 48 36 55 51 98 55;
80 11 21 82 30 89 97 18 18 31 59 76 20 6 94 86 36 10 18 8;
55 61 42 5 93 1 56 57 61 29 84 40 9 89 90 63 15 51 65 57;
52 23 2 91 41 34 28 29 77 81 24 34 86 83 28 88 20 86 5 1;
24 33 90 65 4 98 81 33 73 72 22 29 32 49 25 12 8 51 30 63;
51 53 25 73 36 29 8 93 76 13 44 26 49 73 23 7 56 7 59 92;
38 69 49 91 16 89 25 57 13 40 83 47 69 66 90 71 71 39 32 61;
15 91 29 15 51 63 74 98 52 14 57 41 45 64 17 28 82 59 18 67;
35 55 57 70 68 61 23 15 8 20 7 81 58 27 22 27 43 2 8 36;
80 22 50 5 24 21 64 31 42 71 100 66 11 42 47 41 34 30 20 15;
54 78 51 91 97 3 14 95 7 2 28 19 36 20 59 69 60 28 95 78;
85 35 27 28 93 93 15 81 84 73 10 88 86 96 51 33 51 8 2 23;
30 28 69 40 40 1 35 86 77 71 87 47 41 32 39 82 60 5 31 73;
67 17 96 75 95 64 65 34 16 6 26 95 7 13 66 90 69 80 1 77;
2 90 49 5 36 43 23 73 55 15 19 54 11 50 99 55 82 19 53 9;
64 56 95 31 68 24 51 78 74 59 84 74 10 80 30 1 27 34 56 67;
7 52 27 74 20 17 17 46 78 23 62 27 83 63 77 75 61 59 100 74;
12 39 61 84 49 59 12 100 97 18 22 12 35 85 81 53 62 57 3 79;
83 3 41 34 32 11 53 90 85 80 54 53 94 57 9 31 36 12 7 55;
50 54 33 22 75 57 83 89 22 71 77 98 17 16 55 33 12 36 80 100;
67 28 30 72 8 10 69 95 13 76 33 21 83 21 35 8 45 65 13 39;
15 87 47 32 85 6 4 31 87 6 88 19 95 59 91 29 13 60 25 19;
97 17 29 76 86 4 3 46 98 97 35 17 85 41 90 95 24 6 56 14;
44 28 42 2 16 92 10 14 28 10 92 62 7 3 25 29 4 15 74 23;
10 56 78 34 4 80 92 33 80 31 99 29 40 5 13 80 11 22 72 68;
75 62 54 19 5 4 41 26 84 71 25 15 89 34 96 20 22 89 67 99;
44 33 12 32 98 25 73 41 94 66 62 44 84 1 41 49 44 2 100 88;
46 79 21 71 89 28 33 53 91 76 26 65 79 41 40 89 5 70 39 34;
36 35 71 60 7 26 85 60 80 88 60 51 55 16 11 90 60 31 77 61;
93 64 80 46 7 53 23 84 89 12 32 100 16 11 96 25 89 46 29 17;
23 51 91 3 68 71 64 76 73 85 33 36 91 38 62 92 97 99 40 76;
55 6 70 30 95 66 50 7 13 68 81 7 35 32 1 14 13 2 75 35;
44 66 10 50 18 49 48 76 12 46 17 87 28 54 30 40 92 92 26 18;
69 92 27 11 92 55 51 7 1 30 10 74 75 90 92 44 14 28 12 75;
36 70 65 87 96 45 75 49 35 57 50 92 5 51 53 23 83 98 75 77;
73 10 90 77 34 6 63 74 92 87 56 88 73 73 7 2 29 11 80 24;
35 49 94 4 37 48 82 81 78 6 69 37 59 18 69 29 7 4 21 40;
3 47 95 15 60 7 24 94 22 25 57 37 67 70 83 77 96 38 16 71;
81 90 34 86 81 45 7 61 79 80 12 11 27 41 15 44 1 98 5 93;
98 90 80 14 6 47 71 55 29 32 51 8 94 67 58 55 52 96 56 3;
10 83 23 22 64 24 86 76 76 67 17 1 47 8 29 57 4 68 62 25;
95 28 82 59 10 12 43 45 50 96 75 89 92 52 69 24 36 74 39 7;
59 13 99 99 24 71 58 12 6 55 20 95 57 7 47 54 53 100 30 23;

• Rec41(75×20)

```
31 47 40 39 10 89 75 51 24 4 67 67 98 12 53 71 84 58 2 71;
66 68 98 4 99 76 60 59 7 30 59 80 94 25 9 24 84 83 47 74;
44 36 90 42 44 67 6 60 87 44 64 30 49 35 51 85 34 36 83 82;
70 36 59 25 25 25 30 3 16 81 73 43 95 50 50 53 37 4 51 67;
72 81 70 54 48 33 74 60 76 2 3 66 84 8 97 52 23 96 19 28;
6 5 18 50 41 5 5 31 71 94 97 72 98 69 9 30 48 50 78 88;
52 70 18 63 60 75 15 39 31 68 22 61 40 19 28 60 90 25 47 96;
13 62 14 66 84 4 22 32 9 58 71 55 40 70 22 41 54 9 67 25;
21 91 71 57 18 79 58 84 87 27 59 3 54 56 84 97 70 11 5 74;
29 94 83 61 91 10 20 93 68 50 30 78 60 39 29 75 94 68 55 65;
17 31 67 24 59 34 70 82 82 69 51 80 63 6 1 49 3 41 31 49;
90 89 64 21 1 10 35 1 85 88 14 31 94 66 66 18 4 15 78 97;
77 78 42 50 21 26 58 22 55 79 36 79 5 56 16 6 25 2 63 9;
34 34 94 35 50 92 5 43 98 28 59 99 80 36 70 59 22 38 76 7;
40 19 36 21 41 44 69 90 42 46 51 88 19 94 68 53 75 83 36 41;
99 17 10 90 75 43 68 8 43 69 3 71 55 38 79 85 87 23 85 83;
42 41 84 32 57 75 43 73 81 57 33 95 41 7 94 78 88 70 90 33;
22 26 50 73 15 51 61 83 60 97 83 33 57 60 31 82 34 59 43 13;
20 33 25 63 6 6 14 61 10 100 20 86 8 83 31 70 4 52 61 72;
41 47 51 73 8 3 21 55 85 1 37 75 24 88 63 40 2 33 87 83;
14 81 96 3 25 25 23 39 65 62 50 49 59 10 17 15 59 15 88 46;
93 69 51 76 89 86 86 12 55 68 2 82 93 10 12 45 28 32 55 82;
59 73 34 40 53 20 5 96 85 30 60 72 30 74 89 1 91 47 7 42;
16 79 87 70 98 88 95 41 9 85 16 43 1 13 17 84 50 44 85 97;
94 97 78 63 27 77 98 19 7 93 88 43 59 64 86 15 37 36 56 48;
47 64 98 43 71 98 83 30 83 40 17 45 86 86 14 46 6 57 13 16;
```

附录 2.4 TD 或 TA 类(120 个子问题)

在此问题不直接给出数据，而是给出可产生该 120 个算例的数据的 C 语言程序。该程序可选择机器编号是由 0 还是 1 开始，对于每个算例所产生的数据中，第 1 行包括工件数和机器数，接下来的每行(按工件号由 1 到 n 排)数据表示每个工件各操作的加工机器以及相应的加工时间，其中机器号由 0(或 1)至 $m-1$(或 m)。所测试的计算机/系统/编译器有如下几种：DEC 5000/ Ultrix 4.2/ cc；SUN 10/ Solaris 2/ cc；IBM 6000/ AIX 3.x/ xlc；Atari ST/ TOS/ pure c；IBM-PC/ DOS 6.2/ MS-C；IBM-PC/ Linux/ gcc。必须注意，程序要求 long 和 double 型数据为 64 位(bit)。

程序取条件 VERIFY == 1 and FIRMACIND == 1 时，生成验证文件 ta001，产生的 20 工件 5 机器的数据如下：

```
20 5
1 54 2 79 3 16 4 66 5 58
1 83 2 3 3 89 4 58 5 56
1 15 2 11 3 49 4 31 5 20
1 71 2 99 3 15 4 68 5 85
1 77 2 56 3 89 4 78 5 53
1 36 2 70 3 45 4 91 5 35
1 53 2 99 3 60 4 13 5 53
1 38 2 60 3 23 4 59 5 41
1 27 2 5 3 57 4 49 5 69
1 87 2 56 3 64 4 85 5 13
1 76 2 3 3 7 4 85 5 86
1 91 2 61 3 1 4 9 5 72
1 14 2 73 3 63 4 39 5 8
1 29 2 75 3 41 4 41 5 49
1 12 2 47 3 63 4 56 5 47
1 77 2 14 3 47 4 40 5 87
1 32 2 21 3 26 4 54 5 58
1 87 2 86 3 75 4 77 5 18
1 68 2 5 3 77 4 51 5 68
1 94 2 77 3 40 4 31 5 28
```

产生 TD 或 TA 类 FSP 问题的 C 源程序如下：

```
#define ANSI_C 0      /* 0:K&R 函数类型转换 */
#define VERIFY 0      /* 1:产生验证文件 */
#define FIRMACIND 0 /* 0,1:起始机器编号 */
#include <stdio.h>
#include <math.h>
struct problem {
    long rand_time;       /* 工件的随机种子 */
    short num_jobs;       /* 工件数 */
    short num_mach;       /* 机器数 */
};

#if VERIFY == 1
struct problem S[] = {
    {0, 0, 0},
```

```
  {873654221, 20, 5},
  {0, 0, 0}};

#else /* VERIFY */
struct problem S[] = {
{0, 0, 0},

/* 20工件5机器 */
{873654221, 20, 5},
{379008056, 20, 5},
{1866992158, 20, 5},
{216771124, 20, 5},
{495070989, 20, 5},
{402959317, 20, 5},
{1369363414, 20, 5},
{2021925980, 20, 5},
{573109518, 20, 5},
{88325120, 20, 5},

/* 20工件10机器 */
{587595453, 20, 10},
{1401007982, 20, 10},
{873136276, 20, 10},
{268827376, 20, 10},
{1634173168, 20, 10},
{691823909, 20, 10},
{73807235, 20, 10},
{1273398721, 20, 10},
{2065119309, 20, 10},
{1672900551, 20, 10},

/* 20工件20机器 */
{479340445, 20, 20},
{268827376, 20, 20},
{1958948863, 20, 20},
{918272953, 20, 20},
{555010963, 20, 20},
```

```
{2010851491, 20, 20},
{1519833303, 20, 20},
{1718670931, 20, 20},
{1923497586, 20, 20},
{1829909967, 20, 20},

/* 50工件5机器 */
{1328042058, 50, 5},
{200382020, 50, 5},
{496319842, 50, 5},
{1203030903, 50, 5},
{1730708564, 50, 5},
{450926852, 50, 5},
{1303135678, 50, 5},
{1273398721, 50, 5},
{587288402, 50, 5},
{248421594, 50, 5},

/* 50工件10机器 */
{1958948863, 50, 10},
{575633267, 50, 10},
{655816003, 50, 10},
{1977864101, 50, 10},
{93805469, 50, 10},
{1803345551, 50, 10},
{49612559, 50, 10},
{1899802599, 50, 10},
{2013025619, 50, 10},
{578962478, 50, 10},

/* 50工件20机器 */
{1539989115, 50, 20},
{691823909, 50, 20},
{655816003, 50, 20},
{1315102446, 50, 20},
{1949668355, 50, 20},
{1923497586, 50, 20},
```

```
{1805594913, 50, 20},
{1861070898, 50, 20},
{715643788, 50, 20},
{464843328, 50, 20},

/* 100工件5机器 */
{896678084, 100, 5},
{1179439976, 100, 5},
{1122278347, 100, 5},
{416756875, 100, 5},
{267829958, 100, 5},
{1835213917, 100, 5},
{1328833962, 100, 5},
{1418570761, 100, 5},
{161033112, 100, 5},
{304212574, 100, 5},

/* 100工件10机器 */
{1539989115, 100, 10},
{655816003, 100, 10},
{960914243, 100, 10},
{1915696806, 100, 10},
{2013025619, 100, 10},
{1168140026, 100, 10},
{1923497586, 100, 10},
{167698528, 100, 10},
{1528387973, 100, 10},
{993794175, 100, 10},

/* 100工件20机器 */
{450926852, 100, 20},
{1462772409, 100, 20},
{1021685265, 100, 20},
{83696007, 100, 20},
{508154254, 100, 20},
{1861070898, 100, 20},
{26482542, 100, 20},
```

```
{444956424, 100, 20},
{2115448041, 100, 20},
{118254244, 100, 20},

/* 200 工件 10 机器 */
{471503978, 200, 10},
{1215892992, 200, 10},
{135346136, 200, 10},
{1602504050, 200, 10},
{160037322, 200, 10},
{551454346, 200, 10},
{519485142, 200, 10},
{383947510, 200, 10},
{1968171878, 200, 10},
{540872513, 200, 10},

/* 200 工件 20 机器 */
{2013025619, 200, 20},
{475051709, 200, 20},
{914834335, 200, 20},
{810642687, 200, 20},
{1019331795, 200, 20},
{2056065863, 200, 20},
{1342855162, 200, 20},
{1325809384, 200, 20},
{1988803007, 200, 20},
{765656702, 200, 20},

/* 500 工件 20 机器 */
{1368624604, 500, 20},
{450181436, 500, 20},
{1927888393, 500, 20},
{1759567256, 500, 20},
{606425239, 500, 20},
{19268348, 500, 20},
{1298201670, 500, 20},
{2041736264, 500, 20},
```

```
{379756761, 500, 20},
{28837162, 500, 20},

{0, 0, 0}};
#endif /* VERIFY */

/* 在上下界内均匀产生随机数 */
#if ANSI_C == 1
int unif (long * seed, short low, short high)
#else
short unif (seed, low, high)
long * seed; short low, high;
#endif
{
    static long m = 2147483647, a = 16807, b = 127773, c = 2836;
    double value_0_1;
    long k = * seed / b;
    * seed = a * ( * seed % b) - k * c;
    if( * seed < 0) * seed = * seed + m;
    value_0_1 = * seed / (double) m;
    return (short) (low + floor(value_0_1 * (high - low + 1)));
}

/* 最多支持 500 个工件 20 台机器,对于更大规模的问题需要扩大数组大小 */
short d[21][501];                          /* 加工时间 */
#if ANSI_C == 1
void generate_flow_shop(short p)           /* 根据 S[p]填入 d 和 M 值 */
#else
void generate_flow_shop(p)
short p;
#endif
{
    short i, j;
    long time_seed = S[p].rand_time;
    for(i = 0; i < S[p].num_mach; ++i)             /* 确定随机加工时间 */
        for (j = 0; j < S[p].num_jobs; ++j)        /* 对所有操作 */
            d[i][j] = unif(&time_seed, 1, 99);     /* 各操作的加工时间不超过 99 */
```

```
}

#if ANSI_C == 1
void write_problem(short p)                    /* 写问题 */
#else
void write_problem(p)
short p;
#endif
{
    short i, j;
    FILE *f = NULL;
    char name[6];
    sprintf(name,"ta%03d", p);                 /* 构造文件名 */
    if(! (f = fopen(name,"w"))) {              /* 打开欲写的文件 */
        fprintf(stderr,"file %s error\n", name);
        return;
    }
    fprintf(f,"%d %d\n", S[p].num_jobs, S[p].num_mach);     /* 写第 1 行 */
    for(j = 0; j < S[p].num_jobs; ++j) {
        for(i = 0; i < S[p].num_mach; ++i) {
            fprintf(f,"%2d %2d ", i+FIRMACIND, d[i][j]);   /* 写机器号和加工时间 */
        }
        fprintf(f,"\n");                       /* 写完一个工件后换行 */
    }
    fclose(f);                                 /* 关闭文件 */
}

int main()
{
    short i = 1;
    while(S[i].rand_time) {                    /* I 由 1 直到写完所有问题 */
        generate_flow_shop(i);                 /* 生产问题 i */
        write_problem(i);                      /* 写问题 i */
        ++i;                                   /* 转入下一个问题 */
    }
    return 0;
}
```

参考文献

1. Aarts B J M. 1996. A parallel local search algorithm for the job shop scheduling problem. Master's Thesis. Department of Mathematics and Computing Science, Eindhoven University of technology, The Netherlands
2. Aarts E H L, Van Laarhoven P J M. 1985. Statistical cooling: a general approach to combinatorial optimization problem. Philips Journal of Research, 40: 193~226
3. Adams J, Balas E, Zawack D. 1988. The shifting bottleneck procedure for job-shop scheduling. Management Science, 34(3): 391~401
4. Androulakis I P, Venkatasubramanlan V. 1991. A genetic algorithm framework for process design and optimization. Computers and Chemical Engineering, 15(4): 217~228
5. Angeline P J, Saunders G M and Pollack J B. 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5(1): 54~65
6. Applegate D, Cook W. 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing, 3(2): 149~156
7. Athreya K B, Doss H, Sethuraman J. 1996. On the convergence of the Markov chain simulation method. Annals of Statististics, 24: 69~100
8. Bac F Q, Perov V L. 1993. New evolutionary genetic algorithms for NP-complete combinatorial optimization problems. Biological Cybernetics, 69: 229~234
9. Back T. 1992. The interaction of mutation rate, selection and self-adaptation within genetic algorithms. In: Manner R, Manderick B, ed. Parallel Problem Solving from Nature II. Amstterdam, North Holland: Elsevier, 15~25
10. Baker K R. 1974. Introduction to sequencing and scheduling. Wiley, New York
11. Balas E, Vazacopoulos A. 1998. Guided local search with shifting bottleneck for job-shop scheduling. Management Science, 44(2): 262~275
12. Balas E. 1969. Machine scheduling via disjunctive graphs: an implicit enumeration algorithm. Operations Research, 13: 517~546
13. Baudet P, Azzaro-Pantel C, Pibouleau L, et al. 1999. Coupling of a genetic algorithm and a simulation model for short-term scheduling in a discontinuous fine chemistry plant. Rairo-RO-Operations Research, 33(3): 299~338
14. Ben-Daya M, Al-Fawzan M. 1998. A tabu search approach for the flow shop scheduling problem. European Journal of Operational Research, 109: 88~95.
15. Bierwirth C. 1995. A generalized permutation approach to job-shop scheduling with genetic algorithms. OR Spektrum, 17(2-3): 87~92
16. Bland R G and Shallcross D F. 1989. Large traveling salesman problem arising from experiments in X-ray crystallography: a preliminary report on computation. Operations Research Letters, 8: 125~128

17. Blazewicz J, Domschke W, Pesch E. 1996. The job shop scheduling problem: conventional and new solution techniques. European Journal of Operational Research, 93(1): 1～33

18. Blue J A and Bennett K P. 1998. Hybrid extreme point tabu search. European Journal of Operational Research, 106(2/3): 676～688

19. Boseniuk T, Ebeling W and Engel A. 1987. Boltzmann and Darwin strategies in complex optimization. Physics Letters A, 125(6/7): 307～310

20. Brah S A, Hunsucker J L. 1991. Branch and bound algorithm for flow shop with multiple processors. European Journal of Operational Research, 51: 88～99

21. Brah S A, Loo L L. 1999. Heuristics for scheduling in a flow shop with multiple processors. European Journal of Operational Research, 113: 113～122

22. Brindle A. 1981. Genetic algorithms for function optimization. Ph D Dissertation. Alberta: Univ. of Alberta

23. Brucker P. 1998. Scheduling algorithms. Berllin: Springer-Verlag

24. Burdett R L, Kozan E. 2000. Evolutionary algorithms for flow shop sequencing with non-unique jobs. International Transactions in Operational Research, 7: 401～418

25. Cai L W, Wu Q H, Yong Z Z. 2000. A genetic algorithm with local search for solving job shop problems. Lecture Notes in Computer Science, 1803: 107～116

26. Candido M A B, Khator S K, Barcia R M. 1998. A genetic algorithm based procedure for more realistic job shop scheduling problem. International Journal of Production Research, 36(12): 3437～3457

27. Cardon A, Galinho T, Vacher J P. 2000. Genetic algorithms using multi-objectives in a multi-agent system. Robotics and Autonomic Systems, 33(2-3), 179～190

28. Carlier J, Pinson E. 1989. An algorithm for solving the job shop problem. Management Science, 35(2): 164～176

29. Carlier J. 1978. Ordonnancements a contraintes disjonctives. Operations Research, 12: 333～351

30. Caseau Y, Laburthe F. 1995. Disjunctive scheduling with task intervals. LIENS Technical Report No 95-25, Paris, France

31. Cesari G. 1996. Divide and conquer strategies for parallel TSP heuristics. Computers and Operations Research, 23(7): 681～694

32. Chelouah R, Siarry P. 2000. Tabu search applied to global optimization. European Journal of Operational Research, 123: 256～270

33. Chen C H, Lin J, Yucesan E, Chick S E. 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. Discrete Event Dynamic Systems, 10: 251～270

34. Chen L, Aihara K. 1995. Chaotic simulated annealing by a neural network model with transient chaos. Neural Networks, 8(6): 915～930

35. Chen R. 1980. The basis of immunology. Beijing: Health Press

36. Cheng R W, Gen M, Tsujimura Y. 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms I-representation. Computers and Industrial Engineering, 30 (4): 983～997

37. Cheng R W, Gen M, Tsujimura Y. 1999. A tutorial survey of job-shop scheduling problems using genetic algorithms Ⅱ—hybrid genetic search strategies. Computers and Industrial Engineering, 36(2): 343～364

38. Cheng R, Gen M. 1995. Minmax earliness/ tardness scheduling in identical parallel machine system using genetic algorithm. Computers and Industrial Engineering, 29(1-4): 513～517

39. Cheng R, Gen M. 1997. Parallel machine scheduling problems using memetic algorithms. Computers and Industrial Engineering, 33(3-4): 761～764

40. Cheng T C E. 1989. A heuristic for common due-date assignment and job scheduling on parallels machines. Journal of Operational Research Society, 40(12): 1129～1135

41. Choi C, Lee J. 1998. Chaotic local search algorithm. Artificial Life and Robotics, 2(1): 41～47

42. Choi S H, Ko J W, Manousiouthakis V. 1999. A stochastic approach to global optimization of chemical processes. Computers and Chemical Engineering, 23: 1351～1356

43. Coit D W, Smith A E. 1996. Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. Computers and Operations Research, 23 (6): 515～526

44. Costamagna E, Fanni A and Giacinto G. 1998. A tabu search algorithm for the optimization of telecommunication networks. European Journal of Operational Research, 106(2/3): 357～372

45. Croce F D, Tadei R, Volta G. 1995. A genetic algorithm for the job-shop problem. Computers and Operations Research, 22(1): 15～24

46. Cvijovic D, Klinowski. 1995. Taboo search: an approach to the multiple minima problem. Science, 667: 664～666

47. Dagli C H, Sittisathanchai S. 1993. Genetic neuro-scheduler for job-shop scheduling. Computers and Industrial Engineering, 25(1-4): 267～270

48. Dai L. 1996. Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. Journal of Optimization Theory and Applications, 91(2): 363～388

49. Davis L. 1987. Genetic algorithms and simulated annealing. Los Altos: Morgan Kaufmann Publishers

50. Davis L. 1991. Handbook of genetic algorithm. New York: Van Nostrand Reinhold

51. De Jong K A. 1975. An analysis of the behavior of a class of genetic adaptive systems. Ph D Dissertation. Ann Arber: Univ Microfilms

52. Dell'Amico M and Trubian M. 1993. Applying tabu search to the job shop scheduling problem. Annals of Operations Research, 41: 231～252

53. Demirkol E, Mehta S, Uzsoy R. 1998. Benchmarks for shop scheduling problems. European Journal of Operational Research, 109(1): 137～141

54. Dimopoulos C, Zalzala A M S. 2000. Recent developments in evolutionary computation for

manufacturing optimization: problems, solutions, and comparisons. IEEE Transactions on Evolutionary Computation, 4(2): 93～113

55. Dorigo M, Gambardella L M. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1): 53～66
56. Dorndorf U, Pesch E. 1995. Evolutionary based learning in a job-shop scheduling environment. Computers and Operations Research, 22(1): 25～40
57. Eberhart R C, Dobbins R W. 1991. Designing neural network explanation facilities using genetic algorithms. In: IEEE IJCNN'91, 1758～1763
58. Eiben A E, Aarts E H and Van Hee K M. 1991. Global convergence of genetic algorithms: an infinite markov chain analysis. In: Schwefel H P and Manner R, ed. Parallel Problem Solving from Nature. Heidelberg, Berlin: Springer-Verlag, 4～12
59. Eiben A E, Hinterding R, Michalewicz Z. 1999. Parameter control in evolutionary algorithms. IEEE Transaction on Evolutionary Computation, 3(2): 124～141
60. Eiben A E, Schoenauer M. 2002. Evolutionary computing. Information Processing Letters, 82: 1～6
61. Eilon S and Christofides N. 1969. Distribution management: mathematical modeling and practical analysis. Operational Research Querterly, 20: 309～319
62. Englander A C. 1985. Machine learning of visual recognition using genetic algorithms. In: Proc of the 1st Int Conf on Genetic Algorithms and their Application. Pittsburgh, PA, 197～201
63. Fisher H, Thompson G L. 1963. Probabilistic learning combinations of local job-shop scheduling rules. In: Muth J F and Thompson G L, ed. Industrial Scheduling. Prentice-Hall, Englewood Cliffs, NJ, 225～251
64. Fogel D B. 1993. Applying evolutionary programming to selected traveling salesman problems. Cybernetics and System, 24: 27～36
65. Fogel D B. 1994. An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks, 5(1): 3～14
66. Foo S Y, Takefuji Y, Szu H. 1995. Scaling properties of neural networks for job-shop scheduling. Neurocomputing, 8(1): 79～91
67. Fredman M L, Johnson D S, McGeoch L A, Ostheimer G. 1995. Data structures for traveling salesman. Journal of Algorithms, 18(3): 432～479
68. Garey E L, Johnson D S, Sethi R. 1976. The complexity of flowshop and job-shop scheduling. Mathmatics and Operations Research, 1: 117～129
69. Garey M R and Johnson D S. 1979. Computers and intractability: A guide to the theory of NP-completeness. Freeman
70. Glover F, Kelly J P, Laguna M. 1995. Genetic algorithms and tabu search: hybrids for optimization. Computers and Operations Research, 22(1): 111～134
71. Glover F, Kochenberger G A and Alidaee B. 1998. Adaptive memory tabu search for binary

quadratic programs. Management Science, 44(3): 336～345

72. Glover F, Laguna M. 1992. Tabu search. In: Reeves C R, ed. Modern heuristic techniques for combinatorial problems. Blackwell Scientific Publication, Oxford, 70～150
73. Glover F, Taillard E, Werra D de. 1993. A user's guide to tabu search. Annals of Operations Research, 41: 3～28
74. Glover F. 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research, 13: 533～549
75. Glover F. 1989. Tabu search: Part Ⅰ. ORSA Journal on Computing, 1(3): 190～206
76. Glover F. 1990. Tabu search: Part Ⅱ. ORSA Journal on Computing, 2(1): 4～32
77. Goldberg D E and Segrest P. 1987. Finite markov chain analysis of genetic algorithm. In: Proc of the Second Int Conf on Genetic Algorithms, 1～8
78. Goldberg D E. 1989. Genetic algorithms in search, optimization, and machine learning. MA: Addison-Wesley
79. Grefenstette J J. 1981. Parallel adaptive algorithms for function optimization. technical report, No. CS 81 19. Nashvilli: Vanderbilt University, Computer Science Department
80. Grefenstette J J. 1986. Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 16(1): 122～128
81. Grotschel M and Holland O. 1991. Solution of large-scale symmetric traveling salesman problems. Mathematical programming, 51(2): 141～202
82. Hajek B. 1988. Cooling schedules for optimal annealing. Mathematics of Operations Research, 13(2): 311～329
83. Hajri S, Liouane N, Hammadi S, et al. 2000. A controlled genetic algorithm by fuzzy logic and belief functions for job-shop scheduling. IEEE Transactions on Systems, Man, and Cybernetics -B, 30(5): 812～818
84. Hale J K. 1977. Theory of functional differential equations. New York: Springer-Verlag
85. Hanafi S. 2000. On the convergence of tabu search. Journal of Heuristic, 7: 47～58
86. Hansen P, Mladenovic N. 2001. Variable neighborhood search: principles and applications. European Journal of Operational Research, 130: 449～467
87. Harp S A. 1989. Towards the genetic synthesis of neural networks. In: Proc of 3rd Conf on GA, 360～369
88. Hart E, Ross P, Nelson J A D. 1999. Scheduling chicken catching—an investigation into the success of a genetic algorithm on a real-world scheduling problem. Annals of Operations Research, 92: 363～380
89. Hart E, Ross P. 2000. A systematic investigation of GA performance on job shop scheduling problems. Lecture Notes in Computer Science, 1803: 277～286
90. Hasegawa M, Ikeguchi T, Matozaki T and Aihara K. 1997. An analysis on additive effects of nonlinear dynamics for combinatorial optimization. IEICE Transaction Fundamentals, E80-A (1): 206～213

91. Haupt R. 1989. A survey of priority rule based scheduling. OR Spektrum, 11: 3～16
92. He J, Kang L. 1999. On the convergence rates of genetic algorithms. Theoretical Computer Science, 229: 23～39
93. Heller J. 1960. Some numerical experiments for an M×J flow shop and its decision-theoretical aspects. Operations Research, 8: 178～184
94. Ho Y C, Sreenivas R, Vakili P. 1992. Ordinal optimization of discrete event dynamic systems. Discrete Event Dynamic Systems, 2(2): 61～88
95. Hoitomt D J, Luh P B, Pattipati K R. 1993. A practical approach to job-shop scheduling problems. IEEE Transactions on Robotics and Automation, 9(1): 1～13
96. Holland J H. 1975. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press
97. Hopfield J J and Tank D W. 1985. Neural computation of decision in optimization problem. Biological Cybernetics, 52: 141～152
98. Hopfield J J. 1982. Neural networks and physical systems with emergent collective computational abilities. In: Proc Natl Acad Sci, 79: 2554～2558
99. Hopfield J J. 1984. Neurons with graded response have collective computational properties like those of two-state neuron. In: Proc Natl Acad Sci, 81: 3088～3092
100. Huntley C L and Brown D E. 1996. Parallel genetic algorithms with local search. Computers and Operations Research, 23(6): 559～571
101. Ishibuchi H, Murata T. 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics-C, 28(3): 392～403
102. Ishibuchi H, Yamamoto N, Misaki S, Tanaka H. 1994. Local search algorithms for flow shop scheduling with fuzzy due-date. International Journal of Production Economics, 33(1): 53～66
103. Ishibuchi H, Yamamoto N, Murata T, Tanaka H. 1994. Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems. Fuzzy Set and Systems, 67(1): 81～100
104. Ishii H, Tada M. 1995. Single machine scheduling problem with fuzzy precedence relation. European Journal of Operational Research, 87: 284～288
105. Jain A S and Meeran S. 1999. Deterministic job-shop scheduling: past present and future. European Journal of Operational Research, 113: 390～434
106. James R J W, Buchanan J T. 1998. Performance enhancements to tabu search for the early/tardy scheduling problem. European Journal of Operational Research, 106: 254～265.
107. Jeong I K and Lee J J. 1996. Adaptive simulated annealing genetic algorithm for control application. International Journal of Systems Science, 27(2): 241～253
108. Jiao L, Wang L. 2000. A novel genetic algorithm based on immunity. IEEE Transactions on Systems, Man, and Cybernetics, 30(5): 552～561.

109. Karmarkar U S and Kekre S. 1985. Lot-sizing in multi-item multi-machine job shops. IIE Transactioons, 17: 290～298

110. Keane A J. 1995. Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness. Artificial Intelligence in Engineering, 9(2): 75～83

111. Kelly J D, Davis L. 1991. Hybridizing the genetic algorithm and the K nearest neighbors classification algorithm. In: Proc of 4th Conf on GA. Los Altos, CA: Morgan Kaufmann Publishers, 377～383

112. Khalil H K. 1992. Nonlinear system. New York: Macmillan

113. Khoo L P, Lee S G, Yin X F. 2000. A prototype genetic algorithm-enhanced multi-objective scheduler for manufacturing systems. International Journal of Advanced Manufacturing Technology, 16(2): 131～138

114. Kim G H, Lee G S G. 1998. Genetic reinforcement learning approach to the hetergeneous machine scheduling problem. IEEE Transactions on Robotics and Automation, 14(6): 879～893

115. Kim Y K, Kim Y J and Kim Y. 1996. Genetic algorithms for assembly line balancing with various objectives. Computers and Industrial Engineering, 30(3): 397～409

116. Kirkpatric S, Gelatt C D and Vecchi M P. 1983. Optimization by simulated annealing. Science, 220: 671～680

117. Kirkpatrick S, Toulouse G. 1985. Configuration space analysis of traveling salesman problem. Journal of Physics, 46: 1277～1292

118. Kolahan F and Liang M. 1998. An adaptive TS approach to JIT sequencing with variable processing times and sequence-dependent setups. European Journal of Operational Research, 109(1): 142～159

119. Kolonko M. 1999. Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research, 113(1): 123～136

120. Koonce D A, Tsai S C. 2000. Using data mining to find patterns in genetic algorithm solutions to a job shop scheduling. Computers and Industrial Engineering, 38(3): 361～374

121. Krishnaknmar K. 1989. Micro-genetic algorithms for stationary and non-stationary function optimization. SPIE Intelligent Control and Adaptive Systems, 1196: 289～296

122. Kurbel K, Rohmann T. 1995. A comparison of job-shop scheduling techniques: simulated annealing, genetic algorithms, and mathematical optimization. Wirtschaftsinformatik, 37(6): 581～593

123. Kurbel K, Schneider B and Singh K. 1998. Solving optimization problems by parallel recombinative simulated annealing on a parallel computer—an application to standard cell placement in VLSI design. IEEE Transactions on Systems, Man, and Cybernetics, 28(3): 454～461

124. Kwok T, Smith K A. 1999. Unified framework for chaotic neural-network approaches to combinatorial optimization. IEEE Transactions on Neural Networks, 10(4): 978～981

125. Lagarias J C. 1998. Convergence properties of the Nelder-Mead simplex method in low dimension. SIAM Journal on Optimization, 9(1): 112～158
126. Lageweg B J. 1984. Private communication of with Van Laarhoven et al discussing the achievement of a makespan of 930 for FT10
127. Lawrence S. 1984. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. GSIA, Carnegie-Mellon Univeristy
128. Lawrence S. 1984. Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittshurgh, PA
129. Lee C Y, Piramuthu S, Tsai Y K. 1997. Job shop scheduling with a genetic algorithm and machine learning. International Journal of Production Research, 35(4): 1171～1191
130. Lee I, Sikora R, Shaw M J. 1997. A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes. IEEE Transactions on Systems, Man, and Cybernetics-B, 27(1): 36～54
131. Lee J and Kim Y. 1996. Search heuristics for resource constrained project scheduling. Journal of Operational Research Society, 47: 678～689
132. Lee L H, Lau T W E, Ho Y C. 1999. Explanation of goal softening in ordinal optimization. IEEE Transaction on Automatic Control, 44(1): 94～99
133. Leung Y, Gao Y, Xu Z B. 1997. Degree of population diversity—a perspective on premature convergence in genetic algorithms and its Markov-chain analysis. IEEE Transactions on Neural Networks, 8(5): 1165～1176
134. Li C, Cheng T. 1993. The parallel machine minmax weighted absolute lateness scheduling problem. Naval Research Logistics, 41: 33～46
135. Lin S and Kernighan B W. 1973. An effective heuristic for traveling-salesman problem. Operations Research, 21: 498～516
136. Lin W, Delgado J G, Gause D C, et al. 1995. Hybrid Newton-Raphson genetic algorithm for the traveling salesman problem. Cybernetics and Systems, 26: 387～412
137. Linn R, Zhang W. 1999. Hybrid flow shop scheduling: a survey. Computers and Industrial Engineering, 37: 57～61
138. Luh P B, Chen D, Thakur L S. 1999. An effective approach for job-shop scheduling with uncertain processing requirements. IEEE Transactions on Robotics and Automation, 15(2): 328～339
139. Luh P B, Hoitomt D J. 1993. Scheduling of manufacturing systems using Lagrangian relaxation technique. IEEE Transactions on Automatic Control, 38: 1066～1079
140. Luh P B, Zhao X, Wang Y J, Thakur L S. 2000. Lagrangian relaxation neural networks for job shop scheduling. IEEE Transactions on Robotics and Automation, 16(1): 78～88
141. Lutz C M, Davis K R, Sun M. 1998. Determining buffer location and size in production lines using tabu search. European Journal of Operational Research, 106: 301～316

142. Mahfoud S W and Goldberg D E. 1995. Parallel recombinative simulated annealing: a genetic algorithm. Parallel Computing, 21: 1～28

143. Martin P D. 1996. A time-oriented approach to computing optimal schedules for the job-shop scheduling problem. Ph. D Thesis, School of Operations Research and Industrial Engineering, Cornell University, NY

144. Matsuo H, Suh C J, Sullivan R S. 1988. A controlled search simulated annealing method for the general job-shop scheduling problem. Working paper, 03-04-88, Graduate School of Business, University of Texas at Austin, USA

145. Maturana F, Gu P, Naumann A, et al. 1997. Object-oriented job-shop scheduling using genetic algorithms. Computers in Industry, 32(3): 281～294

146. McMahon G B, Florian M. 1975. On scheduling with ready times and due dates to minimize maximum lateness. Operations Research, 23(3): 475～482

147. Mesghouni K, Pesin P, Trentesaux D, et al. 1999. Hybrid approach to decision-making for job-shop scheduling. Production Planning and Control, 10(7): 690～706

148. Metropolis N, Rosenbluth A W and Rosenbluth M N, et al. 1953. Equations of state calculations by fast computing machines. Journal of Chemical Physics, 21: 1087～1091

149. Michalewicz Z. 1994. Genetic algorithms + data structures = evolution programs. Berlin: Springer-Verlag

150. Mitra D, Romeo F and Sangiovanni-Vincentelli A. 1986. Convergence and finite-time behavior of simulated annealing. Advanced Applied Probability, 18: 747～771

151. Mladenovic N, Hansen P. 1997. Variable neighborhood search. Computers and Operations Research, 24: 1097～1100

152. Muhlenbein H. 1991. Evolution in time and space: the parallel genetic algorithm. In: Raulins G, ed. Foundations of Genetic Algorithms. Morgan Kaufmann

153. Muth J F, Thompson G. 1963. Industrial scheduling. NJ: Prentice Hall, Englewood Cliffs

154. Nakano R and Yamada T. 1991. Conventional genetic algorithm for job shop problems. In: Proc 4th Int Conf Genetic algorithms and their Applications. San Diego, 474～479

155. Nawaz M, Enscore E Jr, Ham I. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, 11(1): 91～95

156. Nowichi E, Smutnicki C. 1996. A fast taboo search algorithm for the job-shop problems. Management Science, 42(6): 797～813

157. Nowicki E, Smutnicki C. 1996. A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research, 91: 160～175

158. Ogbu F A and Smith D K. 1990. The application of the simulated annealing algorithm to the solution of the $n/m/C_{max}$ flow-shop problem. Computers and Operations Research, 17(3): 243～253

159. Ombuki B M, Nakamura M, Onaga K. 1998. An evolutionary scheduling scheme based on gkGA approach to the job shop scheduling problem. IEICE Transactions Fundamental

Electronics Communications and Computer Science, E81A(6): 1063～1071

160. Osman I H, Potts C N. 1989. Simulated annealing for permutation flow-shop scheduling. Omega, 17(6): 551～557

161. Ozdamar L, Birbil S I. 1998. Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. European Journal of Operational Research, 110(3): 525～547

162. Park L J, Park C H. 1995. Genetic algorithm for job-shop scheduling problems based on 2 representational schemes. Electronic Letters, 31(23): 2051～2053

163. Park M and Kim Y. 1998. A systematic procedure for setting parameters in simulated annealing algorithms. Computers and Operations Research, 25(3): 207～217

164. Pham Q T. 1995. Competitive evolution: a natural approach to operator selection. In: X. Yao ed. Progress in Evolutionary Computation, Lecture Notes in Artificial Intelligence. Heidelberg Springer: Vol. 956: 49～60

165. Ponnambalam S G, Aravindan P, Rajesh S V. 2001. A tabu search algorithm for job shop scheduling. International Journal of Advanced Manufacturing Technology, 16(10): 765～771

166. Ponnambalam S G, Jawahar N, Aravindan P. 1999. A simulated annealing algorithm for job shop scheduling. Production Planning and Control, 10(8): 767～777

167. Ponnambalam S G, Jawahar N, Kumar B S. 2002. Estimation of optimum genetic control parameters for job shop scheduling. International Journal of Advanced Manufactuting Technology, 19: 224～234

168. Poths J C, Giddens T D and Yadaw S B. 1994. The development and evaluation for an improved genetic algorithm based on migration and artificial selection. IEEE Transactions on Systems, Man, and Cybernetics, 24(1): 73～86

169. Prins C. 2000. Competitive genetic algorithms for the open-shop scheduling problem. Mathematical Method in Operations Research, 52(3): 389～411

170. Qi J G, Burns G R and Harrison D K. 2000. The application of parallel multipopulation genetic algorithms to dynamic job-shop scheduling. International Journal of Advanced Manufacturing Technology, 16(8): 609～615

171. Qi X and Palmieri F. 1994. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, Part Ⅰ: Basic properties of selection and mutation. IEEE Transactions on Neural Networks, 5(1): 102～119

172. Reeves C R. 1995. A genetic algorithm for flow shop sequencing. Computers and Operations Research, 22(1): 5～13

173. Rego C. 1998. Relaxed tours and path ejections for the traveling salesman problem. European Journal of Operational Research, 106: 522～538

174. Rose C and Yates R D. 1996. Genetic algorithms and call admission to telecommunications networks. Computers and Operations Research, 23(5): 485～499

175. Rosenthal J S. 1995. Minorization conditions and convergence rates for Markov chain Monte

Carlo. J. Amer. Statist. Assoc., 90: 558～566

176. RoyChowdhury P, Singh Y P, Chansarkar R A. 2000. Hybridization of gradient decent algorithms with dynamic tunneling methods for global optimization. IEEE Transactions on Systems, Man, and Cybernetics, 30(3): 384～390
177. Rudolph G. 1994. Convergence properties of canonical genetic algorithms. IEEE Transactions on Neural Network, 5(1): 96～101
178. Sakawa M, Kubota R. 2001. Two-objective fuzzy job shop scheduling through genetic algorithm. Electronics and Communication in Japan-Ⅲ, 84(4): 60～68
179. Sakawa M, Mori T. 1999. An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing tie and fuzzy duedate. Computers and Industrial Engineering, 36: 325～341
180. Santos D L, Hunsucker J L, Deal D E. 1995. Global lower bounds for flow shops with multiple processors. European Journal of Operational Research, 80: 112～120
181. Schraudolph N N, Belew R K. 1992. Dynamic parameter encoding for genetic algorithms. Machine Learning, 9: 9～21
182. Seneta E. 1981. Non-negative matrices and Markov chains. 2nd ed. New York: Springer-Verlag
183. Sexton R S, Alidaee B, Dorsey R E, et al. 1998. Global optimization for artificial neural networks: a tabu search application. European Journal of Operational Research, 106(2/3): 570～584
184. Shi G. 1997. A genetic algorithm applied to a classic job-shop scheduling problem. International Journal of Systems Science, 28(1): 25～32
185. Shi L, Olafsson S, Chen Q. 1999. A new hybrid optimization algorithm. Computers and Industrial Engineering, 36: 409～426
186. Slowinski R, Hapke M. 2000. Scheduling under fuzziness. New York: Physica-Verlag
187. Smith D. 1985. Bin packing with adaptive search. In: Proc of Int Conf on Genetic algorithms and their Applications. 202～206
188. Srinivas M, Patnaik L M. 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transaction on Systems, Man, and Cybernetics, 24(4): 656～667
189. Starkweather T. 1991. A comparison of genetic sequencing operators. In: Proc 4th Int Conf Genetic Algorithms. Los Altos. CA: Morgan Kaufmann Publishers, 69～76
190. Storer R H, Wu S D, Vaccari R. 1992. New search spaces for sequencing problems with applications to job-shop scheduling. Management Science, 38(10): 1495～1509
191. Syswerda G. 1989. Uniform crossover in genetic algorithms. In: 3rd Int Conf on Genetic Algorithms, 2～9
192. Szu H. 1988. Fast TSP algorithm based on binary neuron output and analog input using the zero-diagonal interconnect matrix and necessary and sufficient constraints of the permutation matrix. In: Proc IEEE Int Conf Neural Network, Ⅱ: 259～266

193. Tadei R, Croce F D, Menga G. 1995. Advanced search techniques for the job-shop problem—a comparison. Rairo-RO-Oper Res, 29(2): 179～194
194. Taillard E. 1993. Benchmarks for basic scheduling problems. European Journal Operational Research, 64(2): 278～285
195. Taillard E. 1994. Parallel taboo search techniques for the job-shop scheduling problem. ORSA Journal on Computing, 16(2): 108～117
196. Talbi E G, Hafidi Z, Geib J M. 1998. A parallel adaptive tabu search approach. Parallel Computing, 24: 2003～2019
197. Tezuka M, Hiji M, Miyabayashi K, et al. 2000. A new genetic representation and common cluster crossover for job shop scheduling problems. Lecture Notes in Computer Science, 1803: 297～306
198. Thomsen S. 1997. Metaheuristics combined with branch and bound. Technical Report, Copenhagen Business School, Denmark
199. Vaessens R J M. 1996. Operations research library of problems. Management School, Imperial Colledge, London
200. Van Laarhoven P J M, Aarts E H L, Lenstra J K. 1992. Job shop scheduling by simulated annealing. Operation Research, 40(1): 113～125
201. Varanelli J M and Cohoon J P. 1999. A fast method for generalized starting temperature determination in homogeneous two-stage simulated annealing systems. Computers and Operations Research, 26: 481～503
202. Varela R, Gomez A and Vela C R, et al. 1999. Heuristic generation of the initial population in solving job shop problems by evolutionary strategies. Lecture Notes in Computer Science, 1606: 690～699
203. Verhoeven M G A. 1998. Tabu search for resource-constrained scheduling. European Journal of Operational Research, 106: 266～276
204. Vigo D and Maniezzo V. 1997. A genetic/tabu thresholding hybrid algorithm for the process allocation problem. Journal of Heuristics, 3(2): 91～110
205. Wang B, Wang F, Zhang Q, et al. 2000. Primary structural and quantitative analysis of infeasible solution to job shop scheduling problem. Journal of System Science and System Engineering, 9(2): 164～170
206. Wang L and Wang M. 1997. A hybrid algorithm for earliness-tardiness scheduling problem with sequence dependent setup time. In: Proc. of the 36th IEEE Conf on Decision and Control, 2: 1219～1222
207. Wang L, Li W F, Zheng D Z. 2001. A class of hybrid strategy for adaptive IIR filter design. The 8th International Conference on Neural Information Processing, 85～89
208. Wang L, Li W F, Zheng D Z. 2001. Design higher-order digital differentiator with simulated annealing. The 5th International Conference on Electronic Measurement and Instrument, 883～886

209. Wang L, Smith K. 1998. On chaotic simulated annealing. IEEE Transactions on Neural Networks, 9(4): 716～718

210. Wang L, Zhang L, Zheng D Z. 2002. Ordinal optimization of genetic control parameters for flow shop scheduling. International Journal of Advanced Manufacturing Technology, Vol. 22

211. Wang L, Zheng D Z. 2000. A class of hybrid optimization strategy for parameter estimation and PID tuning. Submitted to Cybernetics and Systems

212. Wang L, Zheng D Z. 2000. An effective modified genetic algorithm for flow-shop scheduling problems. Submitted to Information Sciences

213. Wang L, Zheng D Z. 2000. Global derivative-free training for feed-forward neural networks. In: The 3rd Asian Control Conference, 1570～1575

214. Wang L, Zheng D Z. 2001. An effective optimization strategy for job-shop scheduling problems. Computers and Operations Research, 28(6): 585～596

215. Wang L, Zheng D Z. 2002. A modified evolutionary programming for flow shop scheduling. International Journal of Advanced Manufacturing Technology, Vol. 21

216. Wang L, Zheng D Z. 2002. A modified genetic algorithm for job shop scheduling. International Journal of Advanced Manufacturing Technology, 20(1): 72～76

217. Wang L, Zheng D Z. 2003. An effective hybrid heuristic for flow shop scheduling. International Journal of Advanced Manufacturing Technology, 21(1): 38～44

218. Wang L, Zheng D Z. 2002. An improved evolutionary programming for optimization. The 4th World Congress on Intelligent Control and Robotics, Vol. 3, 1769～1773

219. Wang L, Zheng D Z. 2002. Finite-time performance analysis for genetic algorithm. Progress in Natural Science, 12(12): 940～944

220. Wang W, Brunn P. 2000. An effective genetic algorithm for job shop scheduling. Proceedings of Institute Mechanical Engineering, 214(4): 293～300

221. Wennink M. 1995. Operations research library of problems. Management School, Imperial Colledge, London

222. Whitley D, Gordan V and Mathias K. 1994. Lamarkian evolution, the baldwin effect and function optimization. In: Davidor Y, et al ed. Parallel solving from nature -PPSN Ⅲ. Berlin: Springer, 6～15

223. Widmer M, Hertz A. 1989. A new heuristic method for the flow shop sequencing problem. European Journal of Operational Research, 41: 186～193

224. Wilson V and Pawlay G S. 1988. On the stability of the TSP problem algorithm of Hopfield and Tank. Biological Cybernetics, 58: 63～70

225. Wolpert D H, Macready W G. 1997. No free lunch theorems for optimization. IEEE Transactions Evolutionary Computation, 1(1): 67～82

226. Wong K P and Wong Y W. 1995. Combined genetic algorithm/simulated annealing/fuzzy set approach to short-term generation scheduling with take-or-pay fuel contract. IEEE Transactions on Power Systems, 11(1): 128～136

227. Wu Z M, Zhao C W. 2000. Genetic algorithm approach to job shop scheduling and its use in real-time cases. International Journal Computers Integrated Manufacturing, 13(5): 422～429
228. Xiao W, Hao P, Zhang S, et al. 2000. Hybrid flow shop scheduling using genetic algorithms. In: Proceedings of the 3rd World Congress on Intelligent Control and Automation, 537～541
229. Yao X, Liu Y and Lin G. 1999. Evolutionary programming made faster. IEEE Transactions Evolutionary Computation, 3(2): 82～102
230. Yen J, Liao J C, Bogju L, et al. 1998. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man, and Cybernetics, 28(2): 173～191
231. Zhang L, Wang L, Tang F. Order-based genetic algorithm for flow shop scheduling. In: The First International Conference on Machine Learning and Cybernetics. Beijing, ICMLC'2002
232. Zhou T, Wang L, Sun Z S. 2002. Closed-loop model set validation under a stochastic framework. Automatica, 38(9): 1449～1461
233. 方剑，席裕庚. 1997. 周期性和事件驱动的 Job Shop 滚动调度策略. 控制与决策，12(2)：159～162
234. 王宇，冯允成，韦有双. 1999. Job shop 问题中动画仿真模型的自动生成系统. 北京航空航天大学学报，25(1)：80～84
235. 王凌，王雄，金以慧等. 2001. 基于生命周期和集成平台思想的流程工业 CIMS 体系结构. 化工自动化及仪表，28(1)：1～4，8
236. 王凌，王雄，金以慧. 2001. MES—流程工业 CIMS 发展的关键. 化工自动化及仪表，28(4)：1～5
237. 王凌，王雄. 2002. 流程工业 CIMS 设计的若干要点. 计算机工程与应用，38(10)：50～52,108
238. 王凌，王雄. 2000. 间歇化工过程最优化的研究进展. 清华大学学报，40(S2)：265～269
239. 王凌，闫铭，李清生等. 2001. 高维复杂函数的一类有效混合优化策略. 清华大学学报，41(9)：118～121
240. 王凌，张亮，郑大钟. 2002. 一种广义 TSP 型交通模型及其优化. 计算机工程与应用，38(2)：15～15，42
241. 王凌，张亮，郑大钟. 2002. 仿真优化研究进展. 控制与决策，Vol.18
242. 王凌，张亮，郑大钟. 2002. 随机仿真优化的一类遗传序优化框架. 控制与决策，17(s)：699～702
243. 王凌，李文峰，郑大钟. 2001. 基于一类混合策略的模型参数估计和控制器参数整定研究. 控制与决策，16(5)：530～534
244. 王凌，李文峰，郑大钟. 2001. 基于模拟退火算法的高阶数字微分器设计. 系统工程与电子技术，23(12)：1～3
245. 王凌，李文峰，郑大钟. 2001. 模型参数估计的一类混合策略. 基础自动化，8(1)：5～7，38
246. 王凌，李文峰，郑大钟. 2002. 非最小相位系统的控制器的优化设计. 自动化学报，29(1)：

135～141
247. 王凌，李令莱，郑大钟等. 2001. 非线性时变系统时滞和参数在线联合估计的SMSA方法. 化工自动化及仪表，28(6)：5～9
248. 王凌，李令莱，郑大钟等. 2002. 非线性系统参数估计的一类有效搜索策略. 自动化学报，Vol. 29
249. 王凌，李令莱，郑大钟. 2002. 设计自适应IIR滤波器的一种SMSA策略. 系统工程与电子技术，24(7)：99～102
250. 王凌，郑大钟，李清生. 2001. 混沌优化方法的研究进展. 计算技术与自动化，20(1)：1～5
251. 王凌，郑大钟. 1998. TSP问题次优化求解方法的比较. 控制与决策，13(1)：79～83
252. 王凌，郑大钟. 1998. 一类GASA混合策略及其收敛性研究. 控制与决策，13(6)：699～672
253. 王凌，郑大钟. 1998. 一类批量可变流水线调度问题的研究. 见：CCC'1998，491～494
254. 王凌，郑大钟. 1998. 前向网络的两种混合学习策略. 清华大学学报，38(9)：95～97
255. 王凌，郑大钟. 1999. TSP及其基于Hopfield神经网络优化的研究. 控制与决策，14(6)：669～674
256. 王凌，郑大钟. 1999. 径向基函数网络结构的混合优化策略. 清华大学学报，39(7)：50～53
257. 王凌，郑大钟. 2000. Meta-heuristic算法研究进展. 控制与决策，15(3)：257～262
258. 王凌，郑大钟. 2000. 一种基于退火策略的混沌神经网络优化算法. 控制理论与应用，17(1)：139～142
259. 王凌，郑大钟. 2000. 邻域搜索算法的统一结构和混合优化策略. 清华大学学报，40(9)：125～128
260. 王凌，郑大钟. 2000. 基于Cauchy和Gaussian分布状态发生器的模拟退火算法. 清华大学学报，40(9)：109～112
261. 王凌，郑大钟. 2000. 基于一类非线性特性的FNN训练算法. 控制与决策，15(1)：19～22
262. 王凌，郑大钟. 2000. 基于不同状态发生器的模拟退火算法性能研究. 见：CCC'2000，430～434
263. 王凌，郑大钟. 2001. 一种GASA混合优化策略. 控制理论与应用，18(4)：552～554
264. 王凌，郑大钟. 2001. 一类含同工件流水线调度问题的优化研究. 计算机工程与应用，37(19)：76～78
265. 王凌，郑大钟. 2001. 基于遗传算法的Job Shop调度研究进展. 控制与决策，16(S)：641～646
266. 王凌，郑大钟. 2002. 一类改进进化规划及其性能分析. 计算机工程与应用，38(1)：8～10
267. 王凌，郑大钟. 2002. 几类动态反馈神经网络的稳定性分析. 计算技术与自动化，21(1)：1～6
268. 王凌，郑大钟. 2002. 多目标优化的一类模拟退火算法. 计算机工程与应用，38(8)：4,5,55
269. 王凌，郑大钟. 2002. 求解同顺序加工调度问题的一种改进遗传算法. 系统工程理论与实践，22(6)：74～79
270. 王凌，郑大钟. 2002. 混合优化策略统一结构的探讨. 控制与决策，17(1)：33～36，40
271. 王凌，郑大钟. 1997. 模拟退火算法求解Flow-shop问题的研究. 见：CDC'1997，390～394

272. 王凌. 1997. 随机优化算法和混合优化策略. 北京：清华大学硕士学位论文
273. 王凌. 1999. 混合优化策略和神经网络中若干问题的研究. 北京：清华大学博士学位论文
274. 王凌. 2001. 智能优化算法及其应用. 北京：清华大学 &Springer 出版社
275. 韦有双，杨湘龙，冯允成. 2000. 一种新的求解 Flow Shop 问题的启发式算法. 系统工程理论与实践. 20(9)：41～47
276. 刘民，吴澄，尹文君. 2001. 带特殊工艺约束的并行机器生产线调度问题的一种遗传算法. 自动化学报，27(3)：381～386
277. 刘民，吴澄，蒋新松. 1998. 用遗传算法解决并行多机调度问题. 系统工程理论与实践，18(1)：14～18
278. 刘民，吴澄. 2000. 解决并行多机提前/拖后调度问题的混合遗传算法方法. 自动化学报，26(2)：258～262
279. 刘岩，韩承德，王义和等. 1996. 模拟退火的背景与单调升温的模拟退火算法. 计算机研究与发展，33(1)：4～10
280. 刘勇，康立山，陈毓屏. 1998. 非数值并行算法——遗传算法. 北京：科学出版社
281. 庄镇泉，王煦法，王东生. 1992. 神经网络与神经计算机. 北京：科学出版社
282. 纪树新，钱积新，孙优贤. 1997. 车间作业调度遗传算法中的编码研究. 信息与控制，26(5)：393～400
283. 张讲社，徐宗本，梁怡. 1997. 整体退火遗传算法及其收敛充要条件. 中国科学，E辑，27(2)：154～164
284. 李小平，王风儒，常会友. 1999. 用定界-遗传算法解 Job-shop 调度问题. 电机与控制学报，3(2)：93～96
285. 李令莱，王凌，郑大钟等. 2002. 基于 simplex-annealing 混合方法的模型参数估计. 清华大学学报，42(9)：1207，1208，1213
286. 李兵，蒋慰孙. 1997. 混沌优化方法及其应用. 控制理论与应用，14(4)：613～615
287. 李清生，王凌，郑大钟. 2001. 复杂函数的一种混合优化方法. 见：CDC'2001，350～353
288. 李清生，王凌，郑大钟. 2002. 水轮机系统的控制器整定研究. 基础自动化，8(3)：10～12，50
289. 杨圣祥，汪定伟. 1998. 用遗传算法与自适应神经网络混合方法解 Job-shop 调度问题. 控制与决策，13(S)：402～407
290. 杨行峻，郑君里. 1992. 人工神经网络. 北京：高等教育出版社
291. 陈恩红，刘贵全，蔡庆生. 1998. 基于遗传算法的 Job-shop 调度问题求解方法. 软件学报，9(2)：139～143
292. 孟庆春. 1995. 基因算法及其应用. 济南：山东大学出版社
293. 郑大钟，赵千川. 2001. 离散事件动态系统. 北京：清华大学出版社
294. 郑大钟. 1990. 线性系统理论. 北京：清华大学出版社
295. 郑学哲，王凌等. 1998. 实现 ICF 均匀照明的二元光学器件的混合优化设计. 中国激光 A，25(3)：265～269
296. 姚伟力，杨德礼，胡祥培. 2000. Job-shop 提前/拖期调度问题的研究. 控制与决策，15(3)：

322～324
297. 姚新，陈国良. 1990. 模拟退火算法及其应用. 计算机研究与发展，7：1～6
298. 段黎明，陈进，刘飞. 1998. 基于约束分析的Job Shop调度算法的综述. 重庆大学学报，21(1)：133～138
299. 唐芳，王凌. 2002. 从局部极小到全局最优. 计算机工程与应用，38(6)：56～58
300. 席裕庚，柴天佑，恽为民. 1996. 遗传算法综述. 控制理论与应用，13(6)：697～708
301. 徐心和. 1990. 旅行商问题的一种新解法. 东北大学学报，11(1)：68～73
302. 翁妙凤. 2000. 基于并行进化规划的Job Shop动态调度策略. 小型微型计算机系统，21(6)：620～622
303. 钱晓龙，唐立新，刘文新. 2001. 动态调度的研究方法综述. 控制与决策，16(2)：141～145
304. 顾擎明，宋文忠. 1998. 混合遗传算法在Job-shop调度问题中的应用. 信息与控制，27(5)：369～374
305. 顾擎明，曹丽娟，宋文忠. 1998. 解Job shop调度问题的自适应遗传方法. 控制与决策，13(5)：589～593
306. 康立山，谢云，尤矢勇等. 1998. 非数值并行算法——模拟退火算法. 北京：科学出版社
307. 梁化楼，戴贵亮. 1995. 人工神经网络与遗传算法的结合：进展与展望. 电子学报，23(10)：194～200
308. 彭宏，王兴华. 1997. 具有Elitist选择的遗传算法的收敛速度估计. 科学通报，42(2)：144～147
309. 焦李成. 1992. 神经网络系统理论. 西安：电子科技大学出版社
310. 童刚，李光泉，刘宝坤. 2000. 用遗传算法解决在并行机上带有不同交货期窗口的Job Shop调度问题. 系统工程，18(3)：37～42
311. 童行行，王凌，何京芮. 2002. 旅行商问题基于参考点的相邻插入法及其改进. 计算机工程与应用，38(20)：63～65
312. 董斌，李颖，邵惠鹤等. 1998. 基于遗传算法的一类Job-shop调度. 控制与决策，13(1)：71～74
313. 蓝海，王雄，王凌. 2001. 一类遗传退火算法的函数优化性能分析. 系统仿真学报，13(S)：111～113
314. 蓝海，王雄，王凌. 2002. 复杂函数最优化的改进遗传退火算法. 清华大学学报，42(9)：1237～1240
315. 潘正君，康立山，陈毓屏. 1998. 演化计算. 北京：清华大学出版社